واجهات المستخدم المرئية
Graphical User Interface

Chapter 2

Dr. MamounYounes



واجهات المستخدم المرئية
Graphical User Interface With Windows Forms

- تسمح واجهة المستخدم المرئية GUI للمستخدم التفاعل بشكل رسومي مع البرامج ، ويمنح البرنامج مظهراً واحساساً مميزاً .
- يطلق على واجهة المستخدم التي تستخدم أسلوب الادخال والإخراج المحرفي اسم CHUI أي Character user Interface .
- **أما في تطبيقات windows فإننا نقوم بإنشاء واجهة مستخدم مرئية التي تقدم طريقة إدخال وإخراج المعطيات بواسطة واجهات مرئية .**
- نستخدم تطبيقات Windows Form في بيئة Visual Stduio.NET بدلاً من تطبيقات Console .

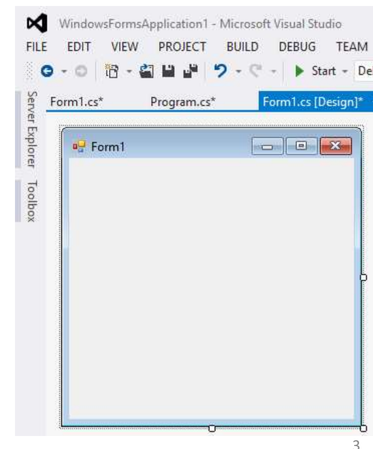
أبسط برنامج Windows

- أبسط برنامج Windows يمكننا إنجازه هو عرض نموذج فارغ كما يلي :

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication21
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }// end form_load
    } // end class
} // end namespace
```



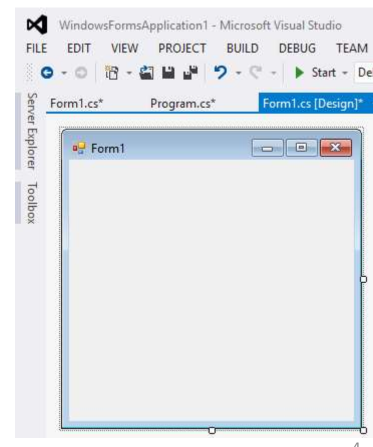
أبسط برنامج Windows

- أبسط برنامج Windows يمكننا إنجازه هو عرض نموذج فارغ كما يلي :

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication21
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }// end form_load
    } // end class
} // end namespace
```

- على الرغم من أن هذه النافذة لا تقدم شيئاً مفيداً ولكنها فعلياً تقدم الكثير .
- **يمكن سحب النافذة** إلى أي مكان من سطح المكتب ، **ويمكن تصغيرها وتكبيرها** من مربعات التحكم الموجودة في الزاوية اليمنى ، **ويمكن إغلاقها** أيضاً .
- **هناك أيقونة في الزاوية اليسارية العليا** يؤدي نقرها إلى عرض قائمة النظام ، ويمكن النقر على حافة النافذة وسحبها لتغيير حجمها .
- **كل هذه الوظائف تم بناءها مسبقاً في النافذة Form والحصول عليها مجاناً .**
- **تتألف النافذة الأساسية من شريط العنوان** title bar **والمحيط** border **والمنطقة التابعة للمستخدم** client area من أجل رسم وعرض واجهة المستخدم الخاصة به .
- لإنشاء نافذة يتطلب التعامل مع الصف Form أو مع صف يرث الصف Form معرف في فضاء الحالة System.Windows.Forms ، ويجب تضمين فضاء الحالة في أعلى البرنامج .

رسائل Windows

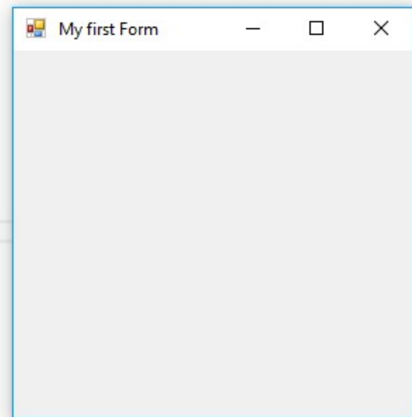
- تستجيب تطبيقات Windows إلى الأحداث المتولدة بنتيجة إدخالات المستخدم ، وقد تتم عن طريق لوحة المفاتيح أو الفأرة أو غيرها من أدوات الإدخال ، فمع كل ضغطة مفتاح أو حركة للفأرة أو نقرة سيتم توليد رسالة حدث Event Message واحدة أو أكثر يتم تسليمها إلى رتل التطبيق الذي يملك النافذة التي كانت نشطة في وقت وقوع الحدث .

وراثة الصف Form

- حتى يقدم الصف Form وظيفة خاصة لتطبيقنا يجب إنشاء صف يرث الصف Form ويقدم كل التوابع وعناصر واجهة المستخدم التي يحتاجها تطبيقنا ، كما هو موضح في المثال التالي :
- وعند تنفيذ البرنامج سوف نحصل على النافذة التالية

بحيث اسم الإطار أو النافذة هو My first Form

أو ما يسمى شريط العنوان للنافذة .



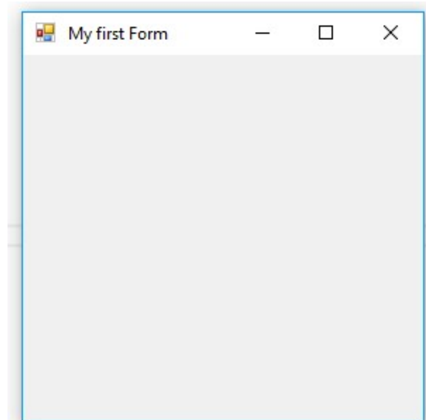
مثال (1)

```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            Text = " My fierst Form ";
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }// end main
    }// end class
}// end nameclace
```



- وسائط معالج الحدث ***Event Handler Parameters***

➢ **يتلقى معالج الحدث وسيطين عند استدعائه :**

✓ اول وسيط (بارامتر) هو مرجع غرض يدعى المرسل sender هو مرجع إلى غرض الذي يولد الحدث event .

✓ الثاني هو مرجع إلى غرض من الصف المشتق EventArgs ويسمى عادة e ويحتوي هذا الغرض على معلومات إضافية حول الحدث الذي ظهر .

✓ EventArgs : هو صف أساسي لكافة الصفوف التي تمثل معلومات الحدث .



عناصر التحكم في GUI
GUI Controls

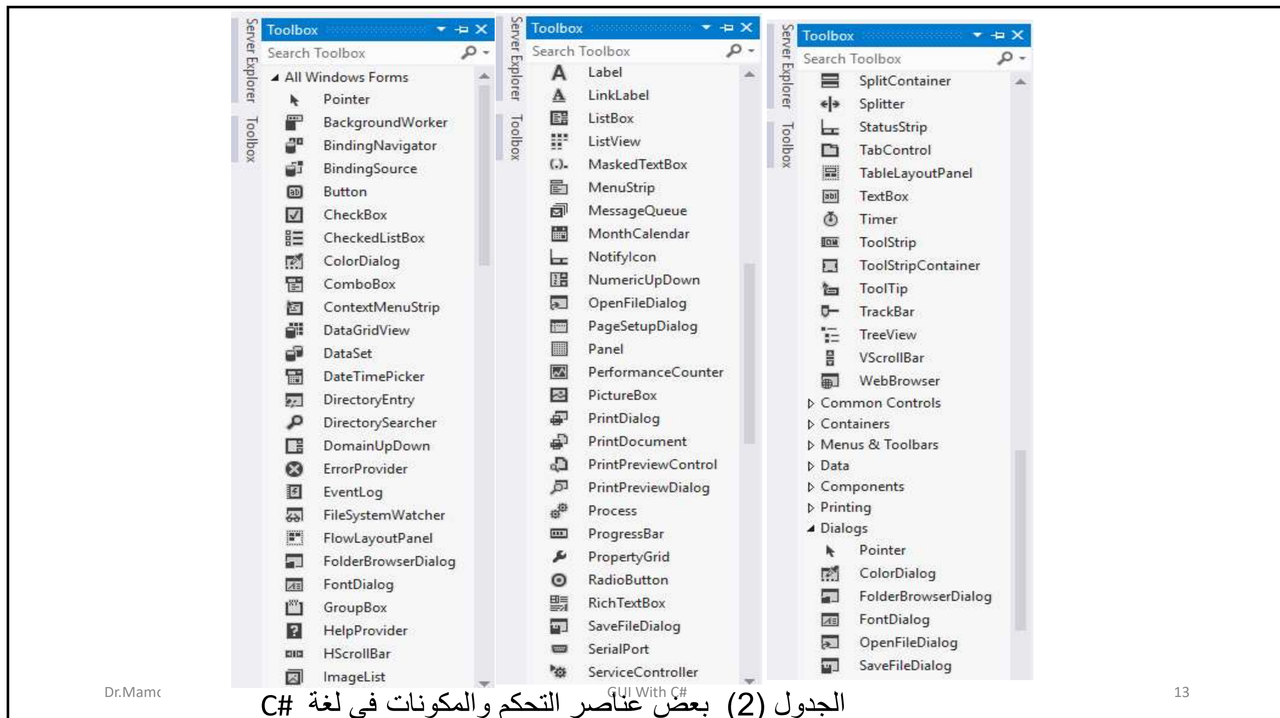
- عنصر التحكم Control هو نوع من النوافذ ويقدم طريقة يتفاعل بها المستخدم مع البرنامج
- **الأزرار Button ومربعات النصوص Textbox** هي أمثلة على عناصر التحكم .
- لكي نتمكن من استخدام عنصر تحكم في تطبيق ما نحتاج إلى اشتقاق نوع عنصر التحكم الذي نريده ، ثم نضيف عنصر التحكم إلى النافذة التي ستحتويه ، ويتم إضافة عنصر التحكم على النافذة بإضافته إلى المجموعة Controls للنافذة .
- هنالك عنصر تحكم بخصائص وتوابع وأحداث يدعمها ، **فالصف Button يمتلك الحدث Click** ، ولمعالجة الحدث Click يجب تسجيل نائب مع هذا الغرض Button .
- يبين الجدول(1) بعض عناصر التحكم الأساسية لـ GUI .
-

Control	Description
Label	Displays *images* or *uneditable text.*
TextBox	Enables the user to *enter data via the keyboard.* It can also be used to *display editable or uneditable text.*
Button	Triggers an *event* when clicked with the mouse.
CheckBox	Specifies an option that can be *selected* (checked) or *unselected* (not checked).
ComboBox	Provides a *drop-down list* of items from which the user can make a *selection* either by clicking an item in the list or by typing in a box.
ListBox	Provides a *list* of items from which the user can make a *selection* by clicking one or more items.
Panel	A *container* in which controls can be placed and organized.
NumericUpDown	Enables the user to select from a *range* of numeric input values.

| Some basic GUI controls.

الجدول(1) بعض عناصر التحكم الأساسية لـ GUI

- **Windows Forms** هي مكتبة يمكن استخدمها لإنشاء واجهات المستخدم المرئية GUI الذي يظهر على سطح المكتب الخاص بنا ، ويمكن أن يكون مربع حوار أو نافذة أو MDI .
- **المكونات Component** هو مثيل instance للصف الذي **ينفذ الواجهة Icomponent interface** التي تحدد السلوكيات التي يجب أن تنفذها المكونات ، مثل كيفية تحميل المكون Component ، ويحتوي عنصر التحكم ، مثل Button or Label، على تمثيل بياني representation graphical في وقت التشغيل ، وتفتقر بعض المكونات إلى الرسوم البيانية (على سبيل المثال ، جهاز ضبط الوقت لمساحة الاسم System.Windows.Forms) .
- يعرض الجدول (2) بعض عناصر التحكم والمكونات في لغة C# لصندوق الأدوات Toolbox.



الجدول (2) بعض عناصر التحكم والمكونات في لغة C#

- **صندوق الأدوات Toolbox**
  - ➢ تعتبر عناصر التحكم هي نوافذ مصممة لإدخال المعطيات وإظهار الخرج .
  - ➢ إن عملية استخدام أي عنصر تحكم يكفي سحب عنصر التحكم من صندوق الأدوات Toolbox إلى النافذة ووضعه في المكان الذي نريد ن وبعدها يتم استخدام النافذة Properties لتحديد خصائص تصميم عنصر التحكم وإضافة أية معالجات أحداث نجدها ضرورية .
- ويبين الجدول(3) الخصائص المشتركة والتوابع والأحداث .
- يبين البرنامج في المثال(2) إنشاء نافذة تتضمن زراً واحداً مع معالج الحدث Click له ، والذي يؤدي إلى عرض مربع رسالة .

Form properties, methods and an event	Description
Common Properties	
AcceptButton	Default Button that's clicked for you when you press *Enter*.
AutoScroll	bool value (false by default) that allows or disallows *scrollbars* when needed.
CancelButton	Button that's clicked when the *Escape* key is pressed.
FormBorderStyle	*Border style* for the Form (Sizable by default).
Font	Font of text displayed on the Form, and the *default font* for controls added to the Form.
Text	Text in the Form's title bar.
Common Methods	
Close	Closes a Form and *releases all resources*, such as the memory used for the Form's contents. A closed Form cannot be reopened.
Hide	Hides a Form, but does *not* destroy the Form or release its resources.
Show	Displays a *hidden* Form.
Common Event	
Load	Occurs before a Form is displayed to the user. You'll learn about events and event-handling in the next section.

Common Form properties, methods and an event.

الجدول(3) الخصائص و التوابع والأحداث العامة

معالجة الأحداث
Event Handling

- عادةً ما يتفاعل المستخدم مع واجهة المستخدم المرئية GUI لتطبيق ما للإشارة إلى المهام التي يجب أن يؤديها التطبيق. **على سبيل المثال** ، عندما تكتب رسالة بريد إلكتروني في تطبيق بريد إلكتروني ، فإن النقر فوق الزر "**إرسال**" يخبر التطبيق بإرسال البريد الإلكتروني إلى عناوين البريد الإلكتروني المحددة. **واجهات المستخدم المرئية GUI مُقادة بالحدث.**
- عندما يتفاعل المستخدم مع مكون من مكونات واجهة المستخدم المرئية GUI، فإن التفاعل - المعروف باسم الحدث Event - يقود البرنامج لتنفيذ مهمة ما.
- تتضمن الأحداث الشائعة (تفاعلات المستخدم) التي قد تتسبب في تنفيذ تطبيق ما مهمة النقر ***click*** على زر Button ، وكتابة نص ***type*** في مربع نص TextBox ، واختيار *select* عنصر من قائمة menu، وإغلاق ***close*** نافذة window وتحريك ***move*** الماوس mouse.

- جميع عناصر واجهة تعامل GUI لها أحداث مرتبطة بها. يمكن أن تشتمل أغراض الأنواع الأخرى أيضًا على أحداث مقترنة أيضًا.
- **تسمى الطريقة التي تقوم بتنفيذ مهمة استجابة لأحد الأحداث بمعالج الحدث** ، وتعرف العملية الشاملة للاستجابة للأحداث باسم معالجة الأحداث.
- **إضافة زر إلى النموذج**

➢ اسحب زر من Toolbox إلى النافذة Form. في صندوق "خصائص" الزر "، قم بتعيين الخاصية (Name) إلى clickButton و الخاصية Text إلى Click Me ستلاحظ أننا نستخدم الاتفاقية التي ينتهي فيها كل اسم متغير ننشئه لعنصر تحكم بنوع التحكم.

➢ على سبيل المثال ، في اسم المتغير clickButton، يكون " **Button** "هو عنصر التحكم و **click** هو معالج الحدث .

- **إضافة معالج أحداث لحدث النقر الخاص بالزر Button's Click Event**

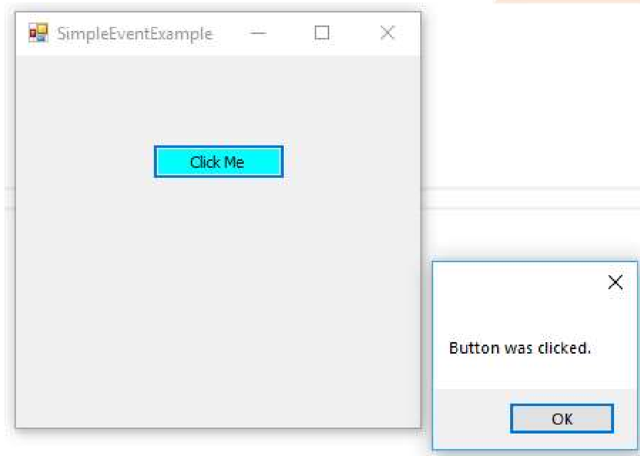
➢ عندما ينقر المستخدم على الزر في هذا المثال ، نريد أن يستجيب التطبيق عن طريق التابع MessageBox.show() للقيام بذلك ، يجب علينا إنشاء معالج أحداث لحدث Click Button ويمكننا إنشاء معالج الأحداث هذا بالنقر المزدوج فوق الزر في النموذج ، الذي يعلن معالج الحدث الفارغ التالي في كود البرنامج .

مثال (2)

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication2
{// Form that shows a simple event handler
    public partial class Form1 : Form
    {  // default constructor
        public Form1()
        {
            InitializeComponent( );
        }// end constructor
        // handles click event of Button clickButton
        private void Form1_Load(object sender, EventArgs e)
        {
            Text = "SimpleEventExample ";
        }
        private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Button was clicked.");
        }
    }//end form
}// end name class
```

ماذا يعني Form1_Load() :
➢ تنفيذ الكود الموجود ضمن القوسين عندما الـ Form1 يحصل لها تحميل Load

الخرج الذي سيظهر بعد تنفيذ البرنامج



- **إعادة تسمية الملف Renaming the Form1.cs File**
  - عند إنشاء نموذج Form يحتوي على زر Button ، أولاً نُنشئ تطبيق Windows Forms جديد ، ثم نقوم بإعادة تسمية الملف Form1.cs إلى SimpleEventExample- Form1.cs في Solution Explorer ثم ننقر فوق النموذج ثم نستخدم الخصائص Properties ، نبحث في الخصائص عن الخاصية Name ثم نعدل اسم الملف .
- **خصائص الـ Form (النافذة)**
  - عند تنفيذ البرنامج سوف تظهر على الشاشة نافذة لها نفس شكل Form ، فتظهر في أعلى ويسار الشاشة ، **حتى نظهرها بالمنتصف** على سبيل المثال نختار من الخصائص الخاصية **StartPosion** فيها عدد من الحالات منها إظهار الشاشة في المنتصف **فنختار الخاصية CenterScreen** .
  - لجعل حجم النموذج Form كبير أو صغير أو عادي نختار من الخصائص الخاصية windowsState.
  - لمعرفة حجم النموذج Form نختار من الخصائص الخاصية Size .
  - الاخفاء لإشارة الـ minimize من النموذج نختار من الخصائص الخاصية MinimizeBox تكون في الحالة العادية true نغيرها إلى false فتختفي إشارة الـ minimize وبنفس الخطوات يمكن إخفاء الـ MaximizeBox عندها لا نستطيع تكبير النافذة أو تصغيرها .
  - في هذه الحالة تم إخفاء أزرار الـ MinimizeBoxو MaximizeBox وليس تم إلغاءها .

- **إعادة تسمية الملف Renaming the Form1.cs File**
  - ➢ عند إنشاء نموذج Form يحتوي على زر Button ، أولاً نُنشئ تطبيق Windows Forms جديد ، ثم نقوم بإعادة تسمية الملف Form1.cs إلى SimpleEventExample- Form1.cs في Solution Explorer ثم ننقر فوق النموذج ثم نستخدم الخصائص Properties ، نبحث في الخصائص عن الخاصية Name ثم نعدل اسم الملف .
- **خصائص الـ Form (النافذة)**
  - ➢ عند تنفيذ البرنامج سوف تظهر على الشاشة نافذة لها نفس شكل Form ، فتظهر في أعلى ويسار الشاشة ، حتى نظهرها بالمنتصف على سبيل المثال نختار من الخصائص الخاصية StartPosion فيها عدد من الحالات منها إظهار الشاشة في المنتصف فنختار الخاصية CenterScreen .
  - ➢ لجعل حجم النموذج Form كبير أو صغير أو عادي نختار من الخصائص الخاصية windowsState.
  - ➢ لمعرفة حجم النموذج Form نختار من الخصائص الخاصية Size .
  - ➢ الاخفاء لإشارة الـ minimize من النموذج نختار من الخصائص الخاصية MinimizeBox تكون في الحالة العادية true نغيرها إلى false فتختفي إشارة الـ minimize وبنفس الخطوات يمكن إخفاء الـ MaximizeBox عندها لا نستطيع تكبير النافذة أو تصغيرها .
  - ➢ في هذه الحالة تم إخفاء أزرار الـ MinimizeBoxو MaximizeBox وليس تم إلغاءها .

- **إعادة تسمية الملف Renaming the Form1.cs File**
  - ➢ عند إنشاء نموذج Form يحتوي على زر Button ، أولاً نُنشئ تطبيق Windows Forms جديد ، ثم نقوم بإعادة تسمية الملف Form1.cs إلى SimpleEventExample- Form1.cs في Solution Explorer ثم ننقر فوق النموذج ثم نستخدم الخصائص Properties ، نبحث في الخصائص عن الخاصية Name ثم نعدل اسم الملف .
- **خصائص الـ Form (النافذة)**
  - ➢ عند تنفيذ البرنامج سوف تظهر على الشاشة نافذة لها نفس شكل Form ، فتظهر في أعلى ويسار الشاشة ، حتى نظهرها بالمنتصف على سبيل المثال نختار من الخصائص الخاصية StartPosion فيها عدد من الحالات منها إظهار الشاشة في المنتصف فنختار الخاصية CenterScreen .
  - ➢ لجعل حجم النموذج Form كبير أو صغير أو عادي نختار من الخصائص الخاصية windowsState.
  - ➢ لمعرفة حجم النموذج Form نختار من الخصائص الخاصية Size .
  - ➢ الاخفاء لإشارة الـ minimize من النموذج نختار من الخصائص الخاصية MinimizeBox تكون في الحالة العادية true نغيرها إلى false فتختفي إشارة الـ minimize وبنفس الخطوات يمكن إخفاء الـ MaximizeBox عندها لا نستطيع تكبير النافذة أو تصغيرها .
  - ➢ في هذه الحالة تم إخفاء أزرار الـ MinimizeBoxو MaximizeBox وليس تم إلغاءها .

- لتغير عنوان النافذة Form نبحث في الخصائص عن خاصية Text ، مثلاً : My Form .
- لإخفاء أو إظهار أيقونة النافذة نبحث في الخصائص عن الخاصية ControlBox ممكن أن تكون true or false .
- لتغير نوع الخط وحجمه داخل النافذة نختار من Properties الخاصية Font .
- لتغير لون خلفية النافذة نختار من الخصائص الخاصية BackColor .
- من أجل تحديد جهة الكتابة على النافذة أو أسماء الأزرار او اللاصقة أو صندوق النص نختار من الخصائص الخاصية RightToLeft أو RightToLeftLayout أي نوع آخر .

- **وسائط معالج الحدث *Event Handler Parameters***

يتلقى معالج الحدث وسيطين عند استدعائه :

- ✓ اول وسيط (بارامتر) هو مرجع غرض يدعى المرسل sender هو مرجع إلى غرض الذي يولد الحدث event .
- ✓ الثاني هو مرجع إلى غرض من الصف المشتق EventArgs ويسمى عادة e ويحتوي هذا الغرض على معلومات إضافية حول الحدث الذي ظهر .
- ✓ EventArgs : هو صف أساسي لكافة الصفوف التي تمثل معلومات الحدث .

- لتغير عنوان النافذة Form نبحث في الخصائص عن خاصية Text ، مثلاً : My Form .
- لإخفاء أو إظهار أيقونة النافذة نبحث في الخصائص عن الخاصية ControlBox ممكن أن تكون true or false .
- لتغير نوع الخط وحجمه داخل النافذة نختار من Properties الخاصية Font .
- لتغير لون خلفية النافذة نختار من الخصائص الخاصية BackColor .
- من أجل تحديد جهة الكتابة على النافذة أو أسماء الأزرار او اللاصقة أو صندوق النص نختار من الخصائص الخاصية RightToLeft أو RightToLeftLayout أي نوع آخر .

- **وسائط معالج الحدث *Event Handler Parameters***

يتلقى معالج الحدث وسيطين عند استدعائه :

- ✓ اول وسيط (بارامتر) هو مرجع غرض يدعى المرسل sender هو مرجع إلى غرض الذي يولد الحدث event .
- ✓ الثاني هو مرجع إلى غرض من الصف المشتق EventArgs ويسمى عادة e ويحتوي هذا الغرض على معلومات إضافية حول الحدث الذي ظهر .
- ✓ EventArgs : هو صف أساسي لكافة الصفوف التي تمثل معلومات الحدث .

الإنابة والأحداث
delegates and events

- يُعرف عنصر التحكم الذي يُنشئ حدثًا باسم مرسل الحدث event sender ، ويُعرف تابع معالجة الحدث معالج الحدث event handler – ويستجيب إلى حدث معين يولده عنصر التحكم.
- عند حدوث الحدث ، يستدعي مرسل الحدث معالج الحدث الخاص به لتنفيذ مهمة (بمعنى "التعامل مع الحدث").
- تسمح لنا NET. آلية التعامل مع الأحداث باختيار الأسماء الخاصة بنا لتوابع التعامل مع الأحداث، ومع ذلك ، يجب أن يصرح كل تابع معالج حدث عن الوسائط المناسبة لتلقي معلومات حول الحدث الذي يتعامل معه.
- نظرًا لأنه يمكننا اختيار أسماء التوابع الخاصة بنا ، لا يمكن لمرسل الحدث مثل زر أن يعرف التابع الذي سيستجيب لهذا الحدث ، لذا ، نحتاج إلى آلية للإشارة إلى تابع معالج الحدث لهذا الحدث .

الصف Form
Form Class

- **يستخدم الصف Form لإنشاء النافذة الرئيسية في التطبيق** ، بحيث يتم انشاء صف مشتق من الصف Form.
- قد نرغب بأن يستجيب التطبيق لأحداث أساسية محددة تحدث أثناء دورة حياة النافذة ، كالحدثين Load و FormClosing اللذان قد نستخدمهما لبدء وتفريغ المعطيات أو المصادر التي تستخدمها النافذة Form .
- نحتاج إلى تسجيل نائب delegate إذا أردنا معالجة حدث خاص بالنافذة Form ، ولتسجيل نائب في نمط التصميم يمكننا النقر على النافذة Form بحيث تظهر خصائصها في الخصائص Properties ليتم عرض قائمة بالأحداث الخاصة بالنافذة Form ، بحيث يتم النقر على الحدث الذي نريد معالجته (نقرة مضاعفة) وسيتولى المصمم الباقي .

- يقوم المصمم بعدها بفتح ملف الشيفرة المصدرية (الكود) في نافذة المحرر مع وضع المؤشر داخل تابع معالج الحدث ، وبالتالي يمكن أن نبدأ مباشرة بكتابة شيفرة (كود) معالجة هذا الحدث .
- يبين المقطع البرمجي التالي معالج حدث نموذجي يعالج الحدث FormClosing ، ومن المتعارف عليه أن نائب معالج الحدث هو أن يعيد void ويقبل وسيطين ، الوسيط الأول هو دائماً مرجع إلى الغرض الذي أطلق الحدث واسمه sender ، أما الوسيط الثاني فسيكون إما مرجع EventArgs ، أو صف مشتق من EventArgs . ومن المتعارف عليه أنه تتم تسمية تابع الحدث باسم الغرض الذي تم تسجيله معه متبوعاً بمحرف الخط السفلي (_) ثم اسم الحدث .


```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
 {
     // ask if they really want to close
      if ( MessageBox.Show(this , “Do you really want to close ?“ ,” Closing “ ,
           MessageBoxbuttons.yesNo , MessageBoxIcon.Qustion ) = = Dialogresult.No )
       {
          e.Concel = true ;
        }
 }
```

- الوسيط الثاني فهو مشتق من FormClosingEventArgs الذي من EventArgs
- يمتلك الغرض من FormClosingEventArgs خاصية هي Cancel يمكننا إعدادها بالقيمة true إذا أردنا إلغاء حدث الإغلاق .
- يعيد التابع MessageBox.Show قيمة الثابت Dialogresult ، حيث يتضمن الثابتDialogresult عدة قيم يمكن أن يُعيد مربعات حوار الرسائل .
- يتضمن مربع حوار الرسالة في حالتنا على زرYes و زر No ، فإذا نقر المستخدم على الزر No فإن التابع MessageBox.Show سيعيد القيمة Dialogresult.No

الصف MassegeBox
MassegeBox Class

- لعرض أي نص ضمن صندوق حوار ، على سبيل المثال الرسالة (message dialog box) ، سنحتاج إلى استخدام التابع Show وظيفته إظهار رسالة ما للمستخدم ، هذا التابع موجود ضمن الصف MessageBox هو أحد صفوف البرمجة المرئية الموجود ضمن C#.
- هو تابع static، ويجب استدعاء هذا التابع باستخدام اسم الصف متبوعاً بنقطة (.) ثم التابع مع الوسائط (البارامترات) ، ويُستخدم لعرض أي نص في صندوق الحوار. حيث يأخذ هذا التابع أربعة(حالياً) وسطاء وهي محملة بشكل زائد .

❖ **الوسيط الأول :** يمثل الرسالة التي ستظهر للمستخدم (Welcome to C#) ، ومن الممكن أن تكون سلسة محرفية تمثل النص الواجب إظهاره .

❖ **الوسيط الثاني:** يمثل الرسالة التي ستظهر على شريط العنوان لصندوق حوار الرسالة ، مثلاً (My First Example) .

❖ **الوسيط الثالث:** فهو يمثل أحد أنواع أزرار حوار الرسالة المبينة في الجدول (4) .

❖ **الوسيط الرابع:** فهو يمثل أحد أنواع أيقونات حوار الرسالة المبينة في الجدول (5) .

❑ أما الوسيطان الأخيران فهما ثوابت يؤثران على شكل مربع الرسالة ، حيث ان قيمة MessageBoxButton.OK تخبر MessageBox ان يقوم بإظهار الزر OK فقط ، بينما قيمة MessageBoxIcon.Information فتخبره ان يظهر أيقونة تبين أنها معلومات مفيدة .

❑ البرنامج التالي يحسب مجموع الأعداد الزوجية من 2 حتى 100 ، ويطبع المجموع على واجهة بيانية (نافذة) ، باستخدام MessageBox.Show .

الجدول(4) – أزرار صندوق حوار الرسالة

MessageBox Buttons	Description
`MessageBoxButton.OK`	Specifies that the dialog should include an **OK** button.
`MessageBoxButton.OKCancel`	Specifies that the dialog should include **OK** and **Cancel** buttons. Warns the user about some condition and allows the user to either continue or cancel an operation.
`MessageBoxButton.YesNo`	Specifies that the dialog should contain **Yes** and **No** buttons. Used to ask the user a question.
`MessageBoxButton.YesNoCancel`	Specifies that the dialog should contain **Yes**, **No** and **Cancel** buttons. Typically used to ask the user a question but still allows the user to cancel the operation.
`MessageBoxButton.RetryCancel`	Specifies that the dialog should contain **Retry** and **Cancel** buttons. Typically used to inform a user about a failed operation and allow the user to retry or cancel the operation.
`MessageBoxButton.AbortRetryIgnore`	Specifies that the dialog should contain **Abort**, **Retry** and **Ignore** buttons. Typically used to inform the user that one of a series of operations has failed and allow the user to abort the series of operations, retry the failed operation or ignore the failed operation and continue.

الجدول(5) – ايقونات صندوق حوار الرسالة

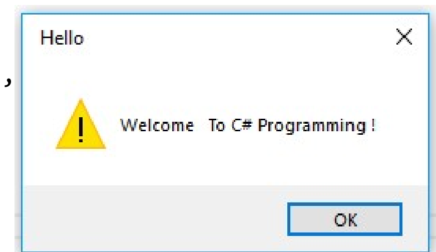
MessageBox Icons	Icon	Description
`MessageBoxIcon.Exclamation`		Specifies an exclamation point icon. Typically used to caution the user against potential problems.
`MessageBoxIcon.Information`		Specifies that the dialog contains an informational message for the user.
`MessageBoxIcon.Question`		Specifies a question mark icon. Typically used in dialogs that ask the user a question.
`MessageBoxIcon.Error`		Specifies a dialog with an **x** in a red circle. Alerts user of errors or important messages.

Icons for message dialogs.

مثال (3)

المثال التالي يوضح خصائص التابع Show من الصف MessageBox.

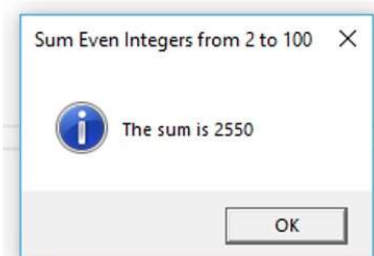
```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication3
{
    public partial class SimpleEventExampleForm1 : Form
    {
        public SimpleEventExampleForm1()
        {
            InitializeComponent();
        }
        private void SimpleEventExampleForm1_Load(object sender, EventArgs e)
        {
            string str = " To C# Programming ! ";
           MessageBox.Show("Welcome  " + str ," Hello",
           MessageBoxButtons.OK,
           MessageBoxIcon. Exclamation);
        }// end form
    }//end class
}// end namespace
```



مثال (4)

```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication3
{
 public partial class SimpleEventExampleForm1 : Form
  {
   public SimpleEventExampleForm1()
   {
     InitializeComponent();
    }
   private void SimpleEventExampleForm1_Load(object sender, EventArgs e)
    {
           int sum = 0;
             for (int number = 2; number <= 100; number += 2)
                 sum += number;
             MessageBox.Show("The sum is " + sum,
             "Sum Even Integers from 2 to 100",
             MessageBoxButtons.OK,
             MessageBoxIcon.Information);
  }// end form
 }// end class
}// end namespace
```



مثال (4)

- البرنـامج التـالي يحسـب الفائـدة المركبـة ، لمستثمر يريـد أن يـودع مبلـغ $1000.00 في حسـاب تـوفير يحقـق فائـدة 5% ، إذا افتراضـنا أن جميـع الفوائـد متبقيـة عنـد الإيـداع ، والمطلوب حساب الفائدة المركبة في نهاية كل عام لمدة عشرة سنوات ، يتم حساب الفائدة المركبة من العلاقة التالية :

$$a = p\,(1 + r)^n$$

- حيث :
- P: هو المبلغ الأصلي المستثمر (أي المبلغ الرئيسي)
- r: هو معدل الفائدة السنوي
- n: هو عدد السنوات
- a: هو المبلغ المودع في نهاية السنة التاسعة.
- ويطبـع الفائـدة والمبلـغ فـي كـل سـنة علـى واجهـة بيانيـة (نافـذة) ، باسـتخدام MessageBox.Show .

مثال (4)

```
// Calculating compound interest.
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication3 {

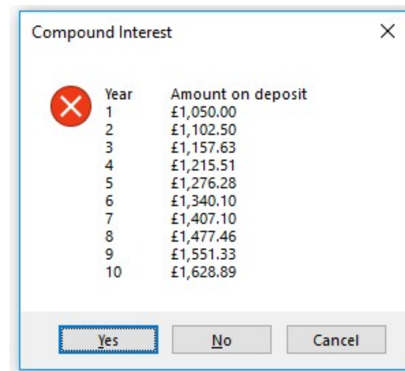
 public partial class SimpleEventExampleForm1 : Form
    {
        public SimpleEventExampleForm1()
        {
            InitializeComponent();
        }

        private void SimpleEventExampleForm1_Load(object sender, EventArgs e)
        {
            decimal amount, principal = (decimal)1000.00;
            double rate = .05;
            string output;
```

```
output = "Year\tAmount on deposit\n";
 for (int year = 1; year <= 10; year++)
 {
    amount = principal * (decimal)Math.Pow(1.0 + rate, year);
    output += year + "\t" + String.Format("{0:C}", amount) + "\n";
            MessageBox.Show(output, "Compound Interest",
          MessageBoxButtons.YesNoCancel, MessageBoxIcon.Stop);

            }// end for

        }// end form
    }//end class
}// end namespace
```



خصائص الصف MassegeBox

- لدينا مشروع project جديد اسمه MyMessageBox ونقوم بمايلي :
  - نبحث في الخصائص عن الخاصية BacColor من أجل جعل خلفية النافذة بيضاء .
  - نبحث في الخصائص عن الخاصية Font من أجل جعل حجم الخط 14 و Bold و Tahoma
  - انشاء زر Button اسمها btnTestMeg ، ومن الخاصية Textنكتب فيه النص My MessageBox ونجعل لون خلفية الزر لون زهري أو أصفر من الخاصية BackColor ونجعل لون الخط أزرق من الخاصية ForeColor.
  - إنشاء لافته Label اسمها lbl ونجعل لون الخط أزرق أو أحمر من الخاصة ForeColor.
  - ننقر على الزر الموجود في النافذة فيظهر التابع التالى :

```
private void btnTestMsg_Click(object sender, EventArgs e)
 {

}
```

- نكتب داخل هذا التابع العبارة `MessageBox.Show(" Hello ");` .

➢نكتب البرنامج بشكل كامل ثم نعيد كتابة التابع بالنسبة لبعض الحالات التي يتميز فيها هذا التابع .

البرنامج

```
using System;
using System.Windows.Forms;

namespace MyMessageBox1
 {
public partial class Form1 : Form
{
public Form1()
 {
   InitializeComponent();
 }

 private void Form1_Load(object sender, EventArgs e)
 {

  }
```

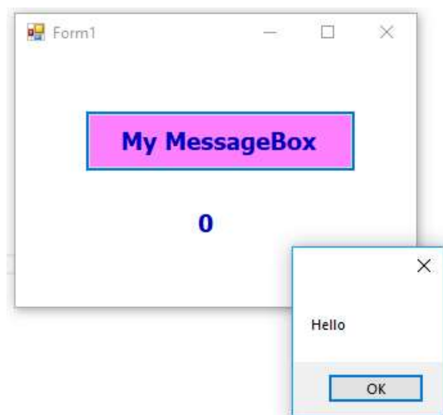


```
private void btnTestMsg_Click(object sender, EventArgs e)
{

   MessageBox.Show(" Hello ");
}
 private void lbl_Click(object sender, EventArgs e)
 {

  }
    }// end class
}// end namespace
```

1- عند إضافة الوسيط الثاني الذي هو عنوان النافذة



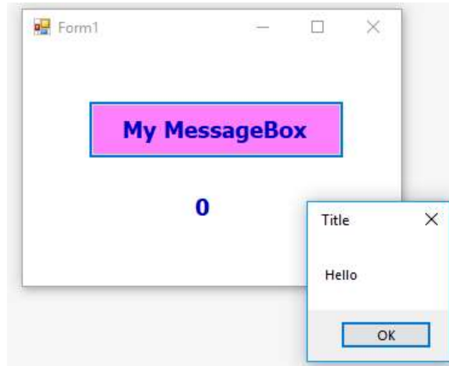
- عند تنفيذ البرنامج يظهر الخرج كما يلي

➢ نجد ظهور العبارة Hello بدون عنوان لشريط الاطار

2- عند إضافة الوسيط الثاني الذي هو عنوان النافذة

```
private void btnTestMsg_Click(object sender, EventArgs e)
{

    MessageBox.Show(" Hello ", " Title  ");
}
```



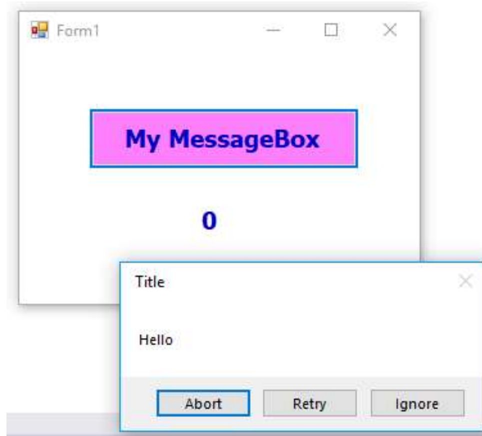
عند تنفيذ البرنامج نحصل على الخرج التالي :

➢ نجد ظهور عنوان شريط الإطار أو النافذة .

3- عند إضافة الوسيط الثالث الذي هو MessageBoxButtons الذي يدل على أنواع الأزرار التي يمكن أن تظهر على النافذة .

```
private void btnTestMsg_Click(object sender, EventArgs e)
{

    MessageBox.Show(" Hello " , " Title  ",MessageBoxButtons.AbortRetryIgnore );
}
```



عند تنفيذ البرنامج نحصل على الخرج التالي :

➢ نجد ظهور ثلاثة أزرار .

بعد إضافة الوسيط الثالث الذي هو `MessageBoxButtons` الذي يدل على أنواع الأزرار التي يمكن أن تظهر على النافذة **نريد إظهار اسم الزر الذي تم الضغط عليه** وذلك كما يلي :

- ننشئ متحول من نوع DialogResult والذي يحتوي على القيمة المعادة من التابع `MessageBox.Show()`.

```
private void btnTestMsg_Click(object sender, EventArgs e)
{

    DialogResult dr;
    dr = MessageBox.Show(" Hello ", " Title  ", MessageBoxButtons.AbortRetryIgnore);
    lbl.Text = dr.ToString();
}
```

عند تنفيذ البرنامج نحصل على الخرج التالي :

- نجد ظهور اسم الزر الذي تم الضغط عليه على نافذة الخرج .



عند الضغط على الزر Ignore

عند الضغط على الزر Retry

عند الضغط على الزر Abort

نريد إظهار اسم الزر الذي تم الضغط عليه باللغة العربية وذلك كما يلي :

- نستخدم تعليمات if – else المتكررة كما هو موضح في البرنامج .

```
private void btnTestMsg_Click(object sender, EventArgs e)
{

    DialogResult dr;
    dr = MessageBox.Show(" Hello ", " Title  ", MessageBoxButtons.AbortRetryIgnore);
    lbl.Text = dr.ToString();
if (dr == DialogResult.Retry)
 {
    lbl.Text = "محاولة  " ;
 }
else if (dr == DialogResult.Ignore)
  {
     lbl.Text = "تجاهل  " ;
   }
else if (dr == DialogResult.Abort)
   {
      lbl.Text = "إلغاء  " ;
   }
}// end  btn
```


عند تنفيذ البرنامج نحصل على الخرج التالي :

- نجد ظهور اسم الزر الذي تم الضغط عليه على نافذة الخرج باللغة العربية .



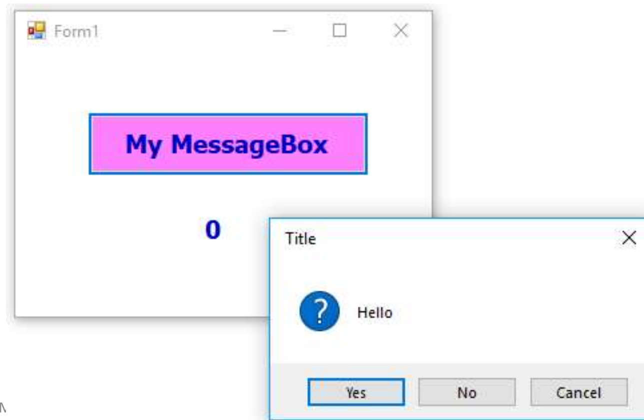
عند الضغط على الزر Ignore　　عند الضغط على الزر Retry　　عند الضغط على الزر Abort

4- عند إضافة الوسيط الرابع الذي هو MessageBoxIcon
الذي يدل على أنواع الأيقونات التي يمكن أن تظهر على النافذة

```
private void btnTestMsg_Click(object sender, EventArgs e)
{
MessageBox.Show(" Hello " , " Title  ", MessageBoxButtons.YesNoCancel,
                 MessageBoxIcon.Question );
}
```

عند تنفيذ البرنامج نحصل على الخرج التالي

➢ نجد ظهور أيقونة على الشاشة .

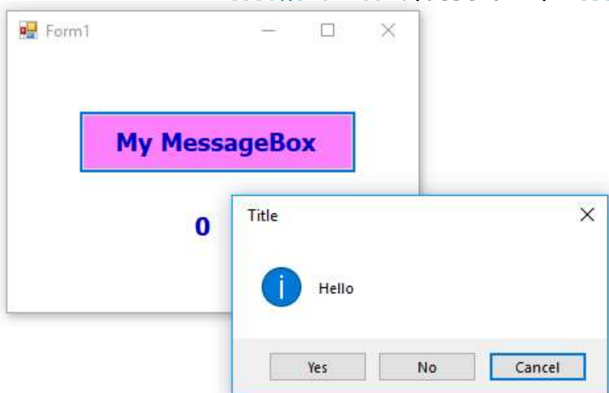


5- عند إضافة الوسيط الخامس الذي هو MessageBoxDefaultButton يحدد الزر الافتراضي الفعال في الحالة الابتدائية عند تنفيذ البرنامج وجعل الزر Button3 هو الافتراضي الذي يظهر على النافذة.

```
private void btnTestMsg_Click(object sender, EventArgs e)
{
MessageBox.Show(" Hello " , " Title  ", MessageBoxButtons.YesNoCancel,
                 MessageBoxIcon.Question  , MessageBoxDefaultButton.Button3 );
}
```

عند تنفيذ البرنامج نحصل على الخرج التالي :

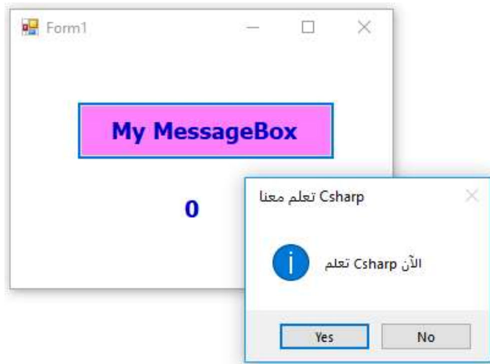
➢ نجد أن الزر Button3 هو الافتراضي على الشاشة .



نريد كتابة الوسيط الأول والثاني باللغة العربية كما يلي :

- الوسيط الأول " تعلم Csharp الآن " أما الوسيط الثاني " تعلم معنا Csharp ".

```
private void btnTestMsg_Click(object sender, EventArgs e)
{
 MessageBox.Show(" الآنCsharp تعلم ,  " تعلم معنا Csharp    ", MessageBoxButtons.YesNo,
                MessageBoxIcon.Information, MessageBoxDefaultButton.Button1);
}
```

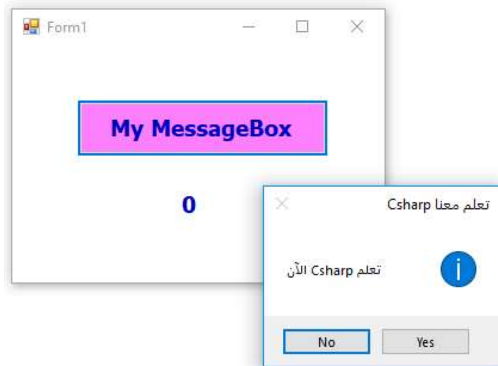


عند تنفيذ البرنامج نحصل على الخرج التالي :

- نلاحظ أننا حصلنا على عبارات غير صحيحه .
- كيف يمكن تصحيحها وذلك بإضافة وسيط آخر من أجل الكتابة باللغة العربية .

6- عند إضافة الوسيط السادس الذي هو MessageBoxOptions
الذي يدل على بعض من Options منها تصحيح اللغة RtlReading

```
private void btnTestMsg_Click(object sender, EventArgs e)
{
 MessageBox.Show(" الآنCsharp تعلم ,  " تعلم معنا Csharp    ", MessageBoxButtons.YesNo,
                MessageBoxIcon.Information, MessageBoxDefaultButton.Button1 ,
                 MessageBoxOptions.RtlReading);
}
```



عند تنفيذ البرنامج نحصل على الخرج التالي :

- نلاحظ أننا حصلنا على عبارات صحيحه .

خصائص عناصر التحكم والتنسيق
Control Properties and Layout

- يتناول هذا القسم الخصائص العامة للعديد من عناصر التحكم. **وتشتق هذه الضوابط من الصف Control**، يوضح الجدول(6) بعض خصائص الصف Controlوالتوابع ، ويمكن تعيين هذه الخصائص لعدد من عناصر التحكم.
- **على سبيل المثال** ، تحدد الخاصية Text النص الذي سيظهر على عنصر التحكم، ويختلف موقع هذا النص حسب عنصر التحكم. في الـ Form، يظهر النص في شريط العنوان ، ولكن يظهر نص الزر على وجهه.
- **تقوم الطريقة Select** بنقل التركيز إلى عنصر التحكم وتجعله عنصراً نشطاً.
- **عند الضغط على المفتاح Tab** في تطبيق Windows Forms تنفيذي ، تتلقى عناصر التحكم التركيز بالترتيب المحدد بواسطة الخاصية TabIndex الخاصة بهم ، تم تعيين هذه الخاصية بواسطة Visual Studio استناداً إلي الترتيب الذي تتم فيه إضافة عناصر التحكم إلى Form ، ولكن يمكننا تغيير ترتيب الجدولة باستخدام VIEW > Tab Order .

Class Control Properties and Methods	Description
Common Properties	
BackColor	Background color of the control.
BackgroundImage	Background image of the control.
Enabled	Whether the control is enabled (i.e., if the user can interact with it). A disabled control will still be displayed, but "grayed-out"—portions of the control will become gray.
Focused	Whether a control has focus. (The control that is currently being used in some way.)
Font	Font used to display control's Text.
ForeColor	Foreground color of the control. This is usually the color used to display the control's Text property.
TabIndex	Tab order of the control. When the *Tab* key is pressed, the focus is moved to controls in increasing tab order. This order can be set by the programmer.
TabStop	If true, user can use the *Tab* key to select the control.
Text	Text associated with the control. The location and appearance varies with the type of control.
TextAlign	The alignment of the text on the control. One of three horizontal positions (left, center or right) and one of three vertical positions (top, middle or bottom).
Visible	Whether the control is visible.
Common Methods	
Focus	Transfers the focus to the control.
Hide	Hides the control (sets Visible to false).
Show	Shows the control (sets Visible to true).

Class Control properties and methods.

الجدول(6) -يوضح بعض خصائص الصف Controlوالتوابع

- تُعد الخاصية TabIndex مفيدة للمستخدمين الذين يدخلون معلومات في العديد من عناصر التحكم ، مثل مجموعة من مربعات النص TextBoxes التي تمثل اسم المستخدم وعنوانه ورقم هاتفه.
- يمكن للمستخدم إدخال المعلومات ، ثم تحديد عنصر التحكم التالي بسرعة عن طريق الضغط على مفتاح Tab.
- تشير الخاصية Enabled إلى إمكانية المستخدم التفاعل مع عنصر تحكم لإنشاء حدث، في كثير من الأحيان ، إذا تم تعطيل disabled عنصر التحكم ، فذلك لأن الخيار غير متوفر للمستخدم في ذلك الوقت.
- على سبيل المثال ، غالبًا ما تعطل تطبيقات محرر النصوص أمر لصق"paste" إلى أن ينسخ المستخدم بعض النصوص ، في معظم الحالات ، يظهر نص عنصر التحكم المُعطل باللون الرمادي (وليس باللون الأسود). يمكنك أيضًا إخفاء عنصر تحكم من المستخدم دون تعطيل عنصر التحكم عن طريق تعيين الخاصية Visible إلى false أو استدعاء الأسلوب Hide. في كل حالة ، لا يزال عنصر التحكم موجودًا ولكنه غير مرئي في Form.

- **Anchoring and Docking**

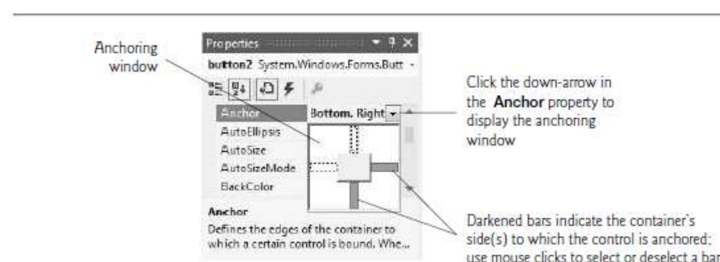
➢يمكنك استخدام التثبيت Anchoring والإرساء Docking لتحديد تنسيق عناصر التحكم داخل حاوية container(مثل Form)، ويتسبب التثبيت Anchoring في بقاء عناصر التحكم على مسافة ثابتة من جانبي الحاوية حتى عند تغيير حجم الحاوية، فالتثبيت Anchoring يعزز خبرة المستخدم.

✓على سبيل المثال ، إذا توقع المستخدم ظهور عنصر تحكم في زاوية معينة من التطبيق ، يضمن التثبيت Anchoring أن عنصر التحكم سيكون دائمًا في هذا الركن — حتى إذا قام المستخدم بتغيير حجم Form.

- يعلق الإرساء Docking التحكم إلى الحاوية بحيث تمتد عنصر التحكم عبر جانب بأكمله أو يملأ منطقة بالكامل.

✓على سبيل المثال ، يمتد زر الارساء إلى أعلى الحاوية عبر الجزء العلوي من تلك الحاوية ، بغض النظر عن عرض الحاوية.

- عندما يتم تغيير حجم النافذة (أو نوع آخر من الحاويات الأصل مثل Panel) ، يتم نقل عناصر التحكم المرتبطة (وربما يتم تغيير حجمها) بحيث لا تختلف المسافة من الجوانب التي ترتكز عليها. بشكل افتراضي ، يتم Anchor معظم عناصر التحكم إلى الزاوية العلوية اليمنى من Form، ولمشاهدة تأثيرات Anchor عنصر التحكم ، نقوم بإنشاء تطبيق Windows Forms بسيط يحتوي على زرين. نقوم بإنشاء تحكمًا واحدًا على الجانبين الأيمن والأسفل عن طريق تعيين خاصية Anchor كما هو موضح في الشكل (1).
- اترك عنصر التحكم الآخر مع Anchor الافتراضي الخاص به في top-left نقوم بتنفيذ التطبيق وبتكبير Form. لاحظ أن الزر المرتبط بالزاوية السفلية اليمنى هو دائمًا على نفس المسافة من الزاوية السفلية اليمنى في Form (الشكل (2)) ، ولكن يبقى عنصر التحكم الآخر على المسافة الأصلية في الزاوية اليسارية العليا من Form.

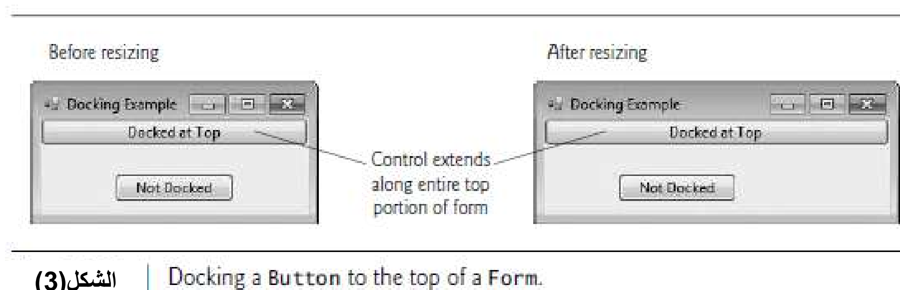


الشكل(1) | Manipulating the **Anchor** property of a control.



الشكل(2) | Anchoring demonstration.

- في بعض الأحيان ، من المرغوب فيه أن يمتد عنصر التحكم على جانب كامل من Form، حتى عند تغيير حجم Form.
- على سبيل المثال ، يجب أن يظل عنصر التحكم مثل شريط الحالة في أسفل Form ، ويتيح الإرساء Dock التحكم لامتداد جانب كامل (يسار أو يمين أو أعلى أو أسفل) للحاوية الرئيسية أو لملء الحاوية بأكملها.
- عندما يتم تغيير حجم عنصر التحكم الرئيسي ، يتم تغيير حجم عنصر التحكم المثبت Dockأيضًا.
- في الشكل (3) ، يتم إرساء Dock زر في أعلى النموذج (يمتد إلى الجزء العلوي) ،وعند تغيير حجم Form، يتم تغيير حجم الزر بما يتناسب مع عرض Form الجديد.
- تحتوي Form على خاصية Padding تحدد المسافة بين عناصر التحكم الراسية Dock وحواف Form ، وتحدد هذه الخاصية أربع قيم (واحد لكل جانب) ، ويتم تعيين كل قيمة إلى 0 بشكل افتراضي ويلخص الجدول(7) بعض خصائص تنسيق التحكم الشائعة.



الشكل(3) | Docking a Button to the top of a Form.

Control layout properties	Description
Anchor	Causes a control to remain at a fixed distance from the side(s) of the container even when the container is resized.
Dock	Allows a control to span one side of its container or to fill the remaining space in the container.
Padding	Sets the space between a container's edges and docked controls. The default is 0, causing the control to appear flush with the container's sides.
Location	Specifies the location (as a set of coordinates) of the upper-left corner of the control, in relation to its container's upper-left corner.
Size	Specifies the size of the control in pixels as a Size object, which has properties Width and Height.
MinimumSize, MaximumSize	Indicates the minimum and maximum size of a Control, respectively.

الجدول(7) | Control layout properties.

الأزرار و اللافتات و مربعات النص

Buttons , Lbles and TextBoxs

والأزرار
Buttons

- **الزر Button هو عنصر تحكم** يقوم المستخدم بالنقر فوقه لتشغيل إجراء محدد أو لتحديد خيار في أحد البرامج.
- كما سنرى ، **يمكن للبرنامج استخدام عدة أنواع من الأزرار** ، مثل مربعات الاختيار checkboxes وأزرار الاختيار radiobuttons .
- **جميع صفوف الأزرار تُشتق من الصف ButtonBase** namespace) System.Windows.Forms) ، التي تحدد ميزات الأزرار الشائعة.
- في هذا القسم ، **نناقش الصف Button**، الذي يمكّن للمستخدم عادةً من إصدار أمر إلى تطبيق ما.
- يعرض الجدول(9)الخصائص العامة والحدث العام في الصف Button.

Button properties and an event	Description
Common Properties	
Text	Specifies the text displayed on the Button face.
FlatStyle	Modifies a Button's appearance—Flat (for the Button to display without a three-dimensional appearance), Popup (for the Button to appear flat until the user moves the mouse pointer over the Button), Standard (three-dimensional) and System, where the Button's appearance is controlled by the operating system. The default value is Standard.
Common Event	
Click	Generated when the user clicks the Button. When you double click a Button in design view, an empty event handler for this event is created.

الجدول(9) | Button properties and an event.

مثال : انشاء زر وعند النقر عليه يقوم بحدث ما

- **المطلوب :** إنشاء زر ، ثم النقر المضاعف على الزر من أجل تغيير خصائصه كما يلي :
  - استخدام الخاصية Name من أجل تسمية التابع الذي ويولد الزر (الاسم btnOk) .
  - استخدام الخاصية Text من أجل الكتابة على وجه الزر (عبارة OK) .
  - استخدام الخاصية BackColor من أجل تلوين الزر .
  - استخدام الخاصية ForeColor من أجل تغيير لون الخط للعبارة التي على وجه الزر .
- **التابع الذي يولد الزر له الشكل التالي :**

```
private void btnOK_Click(object sender, EventArgs e)
  {

   }
```


- **btnOK** هو اسم الزر .
- **Click** هو أمر حدث عند النقر المضاعف عليه يتم إنشاء معالج حدث فارغ ، حتى يتم تنفيذ حدث ما نكتب العبارة التالية داخل تابع الحدث :

```
MessageBox.Show(" Hello  ") ;
```

- **استخدام الخاصة FlatStyle من أجل تعديل مظهر الزر منها :**
  - **Flat** مسطح (لعرض الزر بدون مظهر ثلاثي الأبعاد) .
  - **Popup** منبثق (يظهر الزر مسطحًا حتى يحرك المستخدم مؤشر الماوس فوق الزر) .
  - **Standard** القياسي (ثلاثي الأبعاد).
  - **System** النظام .

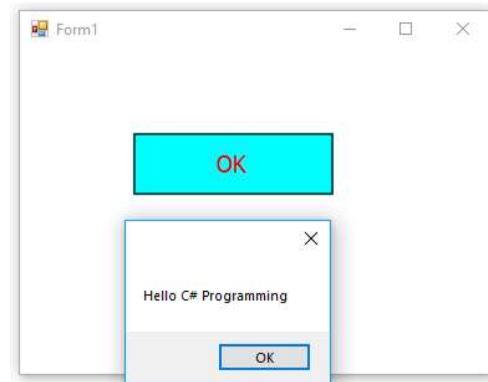
حيث يتم التحكم في مظهر الزر بواسطة نظام التشغيل ،حيث القيمة الافتراضية هي Standard .

البرنامج

```
using System;
using System.Windows.Forms;
namespace Button1 {
public partial class Form1 : Form
{
  public Form1()
  {
     InitializeComponent();
  }

private void Form1_Load(object sender, EventArgs e)
 {

 }
private void btnOK_Click(object sender, EventArgs e)
 {
    MessageBox.Show(" Hello C# Programming ");
  }
 }//end class
}// end namespace
```



اللافتات
Labels

- توفر اللافتات Label معلومات نصية Text (بالإضافة إلى صور اختيارية) ويتم تعريفها باستخدام الصف Label (صف مشتق من الصف Control).
- تعرض اللافتة Label نصاً Text لا يمكن للمستخدم تعديله بشكل مباشر، ويمكن تغيير النص بشكل برمجي عن طريق تعديل خاصية النص Text Label's ، ويوضح الجدول(8) خصائص اللافتات Label الشائعة .
- من خواصها العامة Font ,Text , TextAlign .
- يمكن تغيير نوع الخط ولون الخط باستخدام الخصائص : Font و ForeColor و BackColor.
- يمكن كتابة نص به ، أو إظهار نص عليه باستخدام الخاصية Text .

Common Label properties	Description
Font	The font of the text on the Label.
Text	The text on the Label.
TextAlign	The alignment of the Label's text on the control—horizontally (left, center or right) and vertically (top, middle or bottom). The default is top, left.

الجدول(7) | Common Label properties.

مربعات النص
TextBoxes

- مربع النص هو صف classBox وهو منطقة يمكن عرض أي نص فيها بواسطة برنامج أو يمكن للمستخدم كتابة النص text عبر لوحة المفاتيح.
- كلمة المرور password TextBox : هو TextBoxيخفي المعلومات التي تم إدخالها من قبل المستخدم ، وأثناء قيام المستخدم بكتابة المحارف ، تقوم كلمة المرور TextBox بوضع قناع mask لدخل المستخدم عن طريق عرض محرف كلمة المرور.
- إذا قمنا بتعيين الخاصية UseSystemPasswordChar إلى true، يصبح TextBox كلمة مرور مربع نص.
- غالبًا ما يواجه المستخدمون كلا النوعين من TextBoxes، عند تسجيل الدخول إلى جهاز كمبيوتر أو موقع ويب - يسمح اسم المستخدم TextBox للمستخدمين بإدخال أسماء المستخدمين الخاصة بهم.
- تسمح كلمة المرور TextBoxللمستخدمين بإدخال كلمات المرور الخاصة بهم. ويعرض الجدول(8) الخصائص العامة والحدث العام لـ TextBoxes .

TextBox properties and an event	Description
Common Properties	
AcceptsReturn	If true in a multiline TextBox, pressing *Enter* in the TextBox creates a new line. If false (the default), pressing *Enter* is the same as pressing the default Button on the Form. The default Button is the one assigned to a Form's AcceptButton property.
Multiline	If true, the TextBox can span multiple lines. The default value is false.
ReadOnly	If true, the TextBox has a gray background, and its text cannot be edited. The default value is false.
ScrollBars	For multiline textboxes, this property indicates which scrollbars appear (None—the default, Horizontal, Vertical or Both).
Text	The TextBox's text content.
UseSystem-PasswordChar	When true, the TextBox becomes a password TextBox, and the system-specified character masks each character the user types.
Common Event	
TextChanged	Generated when the text changes in a TextBox (i.e., when the user adds or deletes characters). When you double click the TextBox control in **Design** mode, an empty event handler for this event is generated.

الجدول(8) | TextBox properties and an event.

مثال : انشاء زر ومربع نص وثلاثة لافتات

- أنشاء زر له الخصائص التالية :
  - اسم التابع الذي يولد الزر btnName .
  - لونه زهري لون الخط احمر ومظهره Pooup .
  - العبارة Click Here موجودة على وجه الزر .
- انشاء لافته lable1 فيها العبارة address الخلفية cyan ، أما اللافتة الثانية Lable2 نكتب بها نص هو Name لون الخط أزرق ، والثالثة هي Lable3 نحذف النص الذي بداخلها ولون الخط أزرق .
- أنشاء TextBox مظهره Borderstyle نختار Fixt3D ، لون الخط Pink زهري ، ومن الخاصية TextAlign نختار Center .
- معالج الحدث لمربع النص هو TextChanged يتم توليده عندما يتغير النص في TextBox ، وعند النقر المضاعف عليه يتم إنشاء معالج حدث فارغ .

```
private void btnName_Click(object sender, EventArgs e)
        {
            string strName = txtBxName.Text;
             MessageBox.Show( "Hello "+ strName) ;
           // label3.Text = "مرحبا " + strName;
        }
```

- **معالج الحدث للزر btnName** يحتوي على رسالة ترحيب للمستخدم ، حيث العبارة :

`string strName = txtBxName.Text;`

تعني أن النص الذي يتم كتابته في مربع النص txtBxName سوف يُخزن في المتحول strName ويتم إظهاره بواسطة MessageBox.Show .

- البرنامج التالي يوضح ذلك .

البرنامج

```
using System;
using System.Windows.Forms;
namespace TextBoxLable
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void btnName_Click(object sender, EventArgs e)
        {
            string strName = txtBxName.Text;
             MessageBox.Show( "Hello "+ strName) ;

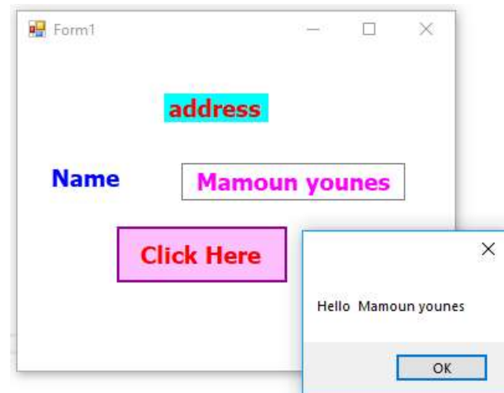
        }
```

```
private void label2_Click(object sender, EventArgs e)
        {

        }

        private void txtBxName_TextChanged(object sender, EventArgs e)
        {

        }
    }//end class
}// end namespace
```



نعيد كتابة البرنامج من أجل اظهار عبارة الترحيب في اللافتة Lable3 المخفية وباللغة العربية .

- يبقى البرنامج كما هو بينما يتغير محتوى تابع الحدث للزر btnName_Click ، كما يلي :

```
private void btnName_Click(object sender, EventArgs e)
        {
            string strName = txtBxName.Text;
            label3.Text =  " مرحبا "+ strName;
        }
```

- العبارة :

```
label3.Text = "  مرحبا "   + strName  ;
```

تعني أظهر الرسالة " مرحبا " والنص الموجود في strName في اللافتة Lable3 المخفية ، وعند تنفيذ البرنامج يظهر الخرج كما هو موضح في الخرج (1) .



الخرج (1)



الخرج (2)

- إذا قمنا بتعيين الخاصية UseSystemPasswordChar إلى true، يصبح TextBox كلمة مرور مربع نص.
- وعند تنفيذ البرنامج يظهر الخرج كما هو موضح في الخرج(2) .

صندوق الاختيار وأزرار الاختيار
CheckBox and RadioButtons

مقدمـــــــة

- تحتوي لغة C # على نوعين من أزرار الحالة state buttons التي يمكن أن تكون في حالات on/off أو حالة true/false ، وهما صناديق الاختيار CheckBoxes و أزرار الاختيار RadioButtons .
- ويتم اشتقاق الصف CheckBoxes و الصف RadioButtons من الصف Button .
- إن صناديق الاختيار وأزرار الاختيار متشابهة وكل منهما يمتلك تسمية وطريقة لتفعيل وتعطيل عنصر التحكم ، كما أن كليهما يخزن قيمة بولانية true أو false تشير إلى ما إذا كان قد تم تفعيل أو تعطيل عنصر التحكم .
- الفارق بينهما هو أنه يمكن وضع أزرار الاختيار ضمن مجموعة بحيث يمكن تفعيل زر اختيار واحد فقط من هذه المجموعة في كل مرة .وهذا يعني أن تفعيل زر اختيار في مجموعة يؤدي حتماً إلى تعطيل كل أزرار الاختيار الأخرى في المجموعة نفسها .

- يمكن تجميع أزرار الاختيار ووضعها ضمن عنصر تحكم حاوي مثل GroupBox ، حيث أن عنصر التحكم GroupBox هو مستطيل مع تسمية في الزاوية اليسارية العليا منه ، ويمكن أن نجد عنصر التحكم في في صندوق الأدوات ToolBox .
- بعد وضع صندوق المجموعة GroupBox في النافذة Form فإنه يقوم بمهمة حاوي لأي عنصر من عناصر التحكم التي نضعه فيه ، فإذا نقرنا على صندوق المجموعة وسحبناه سنلاحظ أن عناصر التحكم التي يحتويها ستنسحب معه .

صندوق الاختيار CheckBox

- صندوق الاختيار CheckBox هو مربع صغير إما فارغ أو يحتوي على علامة اختيار، وعندما يقوم المستخدم بالنقر فوق CheckBox لتحديده ، تظهر علامة اختيار في المربع ، وإذا قام المستخدم بالنقر فوق مربع الاختيار مرة أخرى يتم إلغاء التحديد، وتتم إزالة علامة الاختيار.
- يمكننا أيضًا تكوين CheckBox للتبديل بين ثلاث حالات (محددة وغير محددة وما بينهما) عن طريق تغيير خاصية Three-State من false إلى true ، ويمكن تحديد أي عدد من CheckBoxes في كل مرة.
- يوضح الجدول(10) قائمة بخصائص وأحداث الـ CheckBox العامة .

CheckBox properties and events	Description
Common Properties	
Appearance	By default, this property is set to Normal, and the CheckBox displays as a traditional checkbox. If it's set to Button, the CheckBox displays as a Button that looks pressed when the CheckBox is checked.
Checked	Indicates whether the CheckBox is *checked* (contains a check mark) or unchecked (blank). This property returns a bool value. The default is false (*unchecked*).
CheckState	Indicates whether the CheckBox is *checked* or *unchecked* with a value from the CheckState enumeration (Checked, Unchecked or Indeterminate). Indeterminate is used when it's unclear whether the state should be Checked or Unchecked. When CheckState is set to Indeterminate, the CheckBox is usually shaded.
Text	Specifies the text displayed to the right of the CheckBox.
ThreeState	When this property is true, the CheckBox has three states—*checked*, *unchecked* and *indeterminate*. By default, this property is false and the CheckBox has only two states—*checked* and *unchecked*.
Common Events	
CheckedChanged	Generated when the Checked or CheckState property changes. This is a CheckBox's default event. When a user double clicks the CheckBox control in design view, an empty event handler for this event is generated.
CheckStateChanged	Generated when the Checked or CheckState property changes.

الجدول(10) | CheckBox properties and events.

- **Appearance المظهر الخارجي** : بشكل افتراضي ، يتم تعيين هذه الخاصية إلى عادي Normal، ويعرض CheckBox كصندوق اختيار تقليدي، ويكون له شكل زر عند الضغط عليه .
- **Checked** هذه الخاصية تشير إلى ما إذا كان CheckBox محددًا (يحتوي على علامة اختيار) أو غير محدد (فارغ)، وهذه الخاصية بإرجاع قيمة bool ، وبشكل افتراضي هو false غير محدد .
- **CheckState** هذه الخاصية تشير إلى ما إذا كان CheckBox محددًا أو غير محدداً مع قيمة من ثوابت الـ CheckState هي :
  - ✓ Unchecked غير محدد .
  - ✓ Checked محدد .
  - ✓أو Indeterminate ما بينهما .
- **المثال التالي يوضح الـ CheckState في حالة false وفي حالة true .**

مثال : انشاء زر ومربع نص وصندوق اختيار (false)
ThreeState في الحالة false

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication5  {
    public partial class Form1 : Form {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {

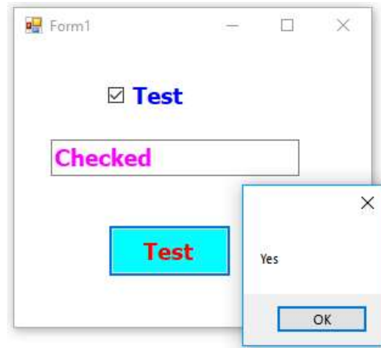
        }
        private void button1_Click(object sender, EventArgs e)
        {
            textBox1.Text = checkBox1.CheckState.ToString();

              if ( checkBox1.Checked )
               MessageBox.Show(" Yes ");
            else
                MessageBox.Show(" No ");
         }
```

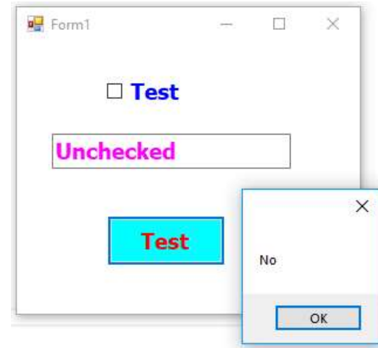
```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {

        }
        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }
    }//end class
}// end namespace
```



عند الضغط على صندوق الاختيار



الحالة الافتراضية لصندوق الاختيار

البرنامج

ThreeState في الحالة true

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication5
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {

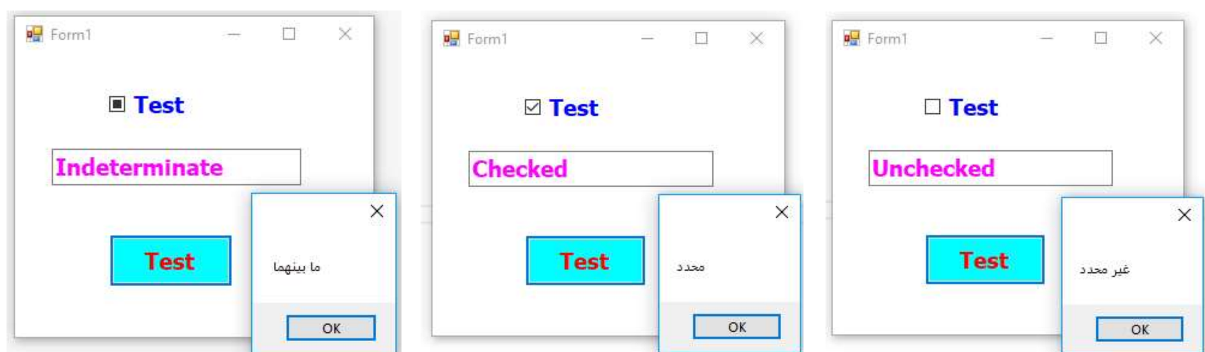
        } private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {

        }
        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }
```

```
private void button1_Click(object sender, EventArgs e)
     {
            textBox1.Text = checkBox1.CheckState.ToString();

               if (checkBox1.CheckState.ToString() == " Unchecked ")
              {
                  MessageBox.Show("  غير محدد “ );
              }
              else if (checkBox1.CheckState.ToString() == " Checked ")
              {
                  MessageBox.Show("  محدد “ ) ;
              }
              else
              {
                  MessageBox.Show("  ما بينهما “ ) ;
              }
        }
    }//end class
}// end namespace
```

عند الضغط للمرة الثانية أي ما بينهما

عند الضغط أول مرة أي محددة

الحالة الافتراضية أي غير محددة

مثال : انشاء لافتة وصندوقين اختيار من أجل كتابة نص في اللافتة ثم استخدام صندوق اختيار لجعل الخط Bold والآخر لجعل الخط Italic

- **يحتوي boldCheckBox1** على الخاصية Textالخاص به لتعيين غامق Bold ، **ويحتوي italicCheckBox2** على الخاصية Textالخاص به لتعيين مائل Italic ، ويتم تعيين الخاصية Textالخاصة outputLabel1 إلى مشاهدة تغيير نمط الخط.
- بعد إنشاء عناصر التحكم ، **نعرّف معالجات الأحداث الخاصة بهم**، ويؤدي النقر المزدوج فوق CheckBoxes في وقت التصميم إلى إنشاء معالجات أحداث CheckedChanged فارغة.
- **نغير نمط الخط في Label**، من الخاصية Font الخاصة به إلى غرض **new Font** ، حيث الصف Font موجود في الفضاء System.Drawing ، ويأخذ باني الصف الذي نستخدمه هنا وسيطين ، الوسيط الأول هو نمط الخط الحالي والثاني هو النمط الجديد كوسائط new style وهما كما يلي :

- **يستخدم الوسيط الأول outputLabel1.Font** اسم وحجم الخط الأصلي لمخرجات outputLabel1 (النمط الافتراضي) ، **بينما الوسيط الثاني** يتم به تحديد النمط الجديد مع عضو في الـ FontStyle، والذي يحتوي على :

Regular, Bold, Italic, Strikeoutand Underline

- **وتكون الخاصية Style لغرض Font للقراءة فقط** ، بحيث يمكن تعيينها فقط عند إنشاء غرض من الصف Font .
- **في السطر 10**، إذا كان outputLabel.Font.Style غامقًا Bold في الأصل ، فلن يكون النمط الناتج Bold ، وإذا كان النص مائلاً في الأصل Italic، يكون النمط الناتج عريضًا ومائلًا ، وليس عريضًا فقط.
- **ينطبق الشيء نفسه على FontStyle.Italic في السطر 11.**
- لأن العملية ^ هي عملة منطقية EXOR على مستوى البت .

البرنامج

```
using System;
using System.Drawing;

using System.Windows.Forms;

namespace WindowsFormsApplication6
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
             outputlabel1.Font = new Font(outputlabel1.Font,
                                             outputlabel1.Font.Style ^ FontStyle.Bold); // 10
        }
```

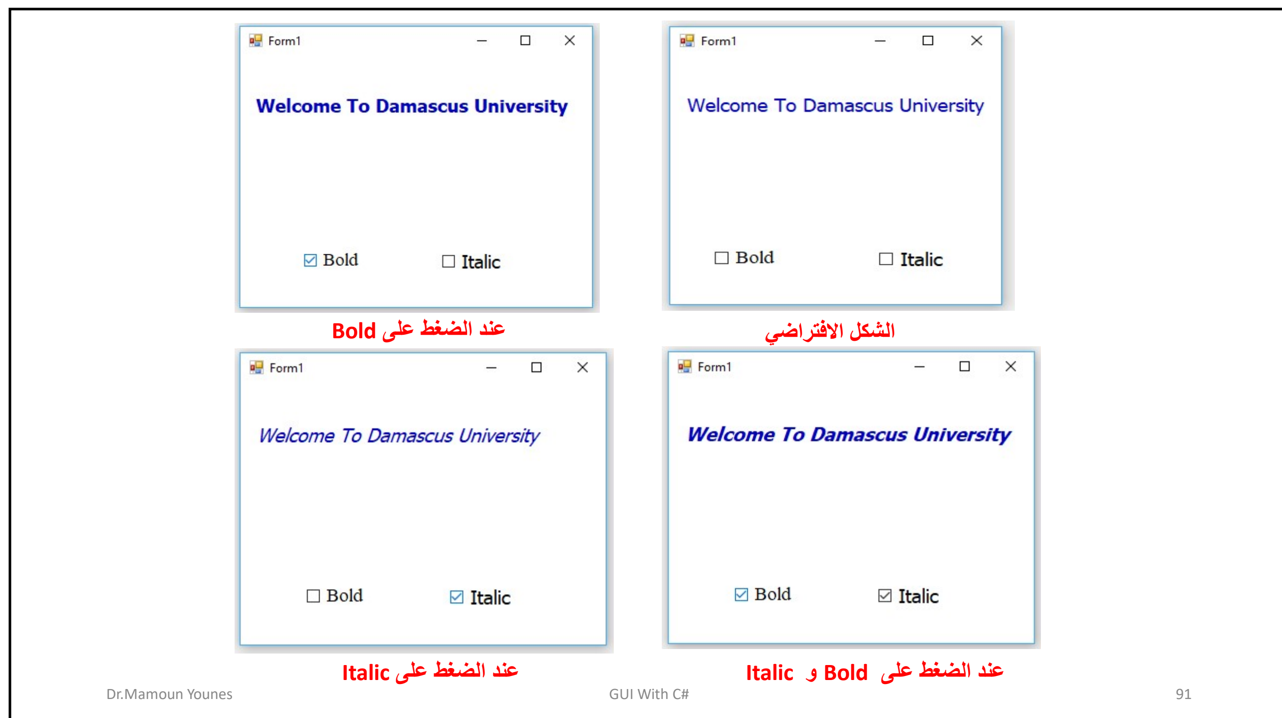


```
private void ItaliccheckBox2_CheckedChanged(object sender, EventArgs e)
        {
          outputlabel1.Font = new Font(outputlabel1.Font,
                                       outputlabel1.Font.Style ^ FontStyle.Italic); // 11
        }

private void outputlabel1_Click(object sender, EventArgs e)
        {

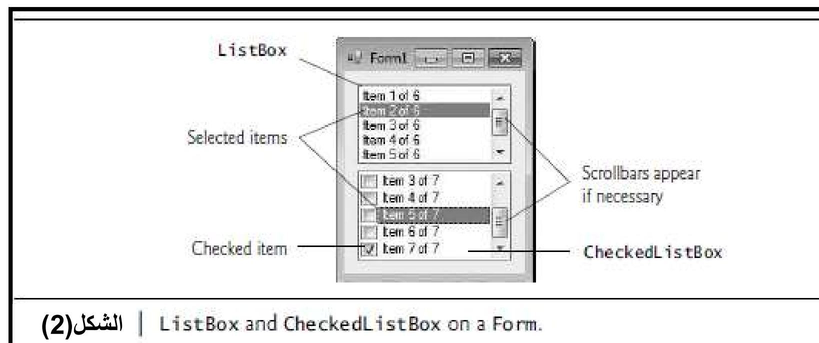
        }

}//end class
}// end namespace
//Welcome To Damascus University
```



صندوق القائمة ListBox

- **يسمح عنصر التحكم ListBox** للمستخدم بعرض وتحديد عناصر متعددة في قائمة.
- **صندوق القائمة ListBox هي أغراض GUI ثابتة Static**، مما يعني أنه لا يمكن للمستخدمين تحرير قائمة العناصر مباشرة ، ويمكن تزويد المستخدم بـ TextBoxes و Buttons لتعيين العناصر المراد إضافتها إلى القائمة ، ولكن يجب تنفيذ الإضافات الفعلية في الكود.
- **يتم توسيع عنصر التحكم CheckedList-Box في ListBox** بتضمين CheckBoxesبجانب كل عنصر في القائمة، مما يتيح ذلك للمستخدمين إجراء عمليات فحص للعناصر المتعددة مرة واحدة ، كما هو ممكن مع عناصر التحكم في CheckBox
- **يمكن للمستخدمين أيضًا تحديد عناصر متعددة من ListBoxعن طريق تعيين الخاصية SelectionMode في ListBox**، ويعرض الشكل (2) الـ ListBox و CheckedListBox وتظهر أشرطة التمرير إذا تجاوز عدد العناصر مساحة العرض في ListBox .



الشكل(2) | ListBox and CheckedListBox on a Form.

- يعرض الجدول(14)الخصائص والتوابع العامة في ListBox والحدث العام.
- تحدد الخاصية SelectionModeعدد العناصر التي يمكن تحديدها ، وتحتوي هذه الخاصية على القيم الممكنة None و One و MultiSimple و MultiExtended من قائمة SelectionMode ، ويتم توضيح الاختلافات بين هذه الإعدادات في الجدول(13)، ويحدث الحدث SelectedIndexChanged عندما يحدد المستخدم عنصرًا جديدًا.

ListBox properties, methods and an event	Description
SelectedIndex	Returns the index of the selected item. If no items have been selected, the property returns -1. If the user selects multiple items, this property returns only one of the selected indices. If multiple items are selected, use property SelectedIndices.
SelectedIndices	Returns a collection containing the indices for all selected items.
SelectedItem	Returns a reference to the selected item. If multiple items are selected, it can return any of the selected items.
SelectedItems	Returns a collection of the selected item(s).
SelectionMode	Determines the number of items that can be selected and the means through which multiple items can be selected. Values None, One (the default), MultiSimple (multiple selection allowed) or MultiExtended (multiple selection allowed using a combination of arrow keys or mouse clicks and *Shift* and *Ctrl* keys).
Sorted	Indicates whether items are sorted *alphabetically*. Setting this property's value to true sorts the items. The default value is false.
Common Methods	
ClearSelected	Deselects every item.
GetSelected	Returns true if the item at the specified index is selected.
Common Event	
SelectedIndexChanged	Generated when the selected index changes. This is the default event when the control is double clicked in the designer.

الجدول(13) | ListBox properties, methods and an event. (Part 2 of 2.)

- **يحتوي كل من ListBox و CheckedListBox** على الخصائص Items و SelectedItem و SelectedIndex ، تُرجع الخاصية Items مجموعة من عناصر القائمة، ويُعد محرر النصوص *Collections* طريقة شائعة لإدارة قوائم الأغراض في NET framework.
- **تستخدم العديد من مكونات (NET GUI.)** مجموعات لعرض قوائم الأغراض الداخلية (على سبيل المثال ، العناصر الموجودة في ListBox) .
- يتم تمثيل المجموعة التي يتم إرجاعها بواسطة الخاصية Items كغرض من نوع ListBoxObject.Collection .
- **الخاصية Selected-Item تقوم** بإرجاع العنصر المحدد حاليًا في ListBox ، إذا أراد **المستخدم تحديد عناصر متعددة ، يتم استخدام المجموعة SelectedItems** لإرجاع كافة العناصر المحددة كقائمة ListBox.SelectedObject-Colection ، **الخاصية SelectedIndex** تقوم بإرجاع فهرس العنصر المحدد - اذا كان هناك أكثر من واحد ، استخدم الخاصية SelectedIndices، والتي تقوم بإرجاع ListBox.Selected-IndexColection

إذا لم يتم تحديد أي عناصر ، فستعرض الخاصية SelectedIndex وإرجاع (1-) .

- يأخذ التابع GetSelected فهرس index ويعيد true إذا تم تحديد العنصر المطابق .

- **إضافة عناصر إلى ListBoxes و CheckedListBoxes**

➢ لإضافة عناصر إلى ListBox أو CheckedListBox، يجب إضافة أغراض إلى مجموعة العناصر الخاصة به.، ويمكن تحقيق ذلك عن طريق استدعاء التابع Add لإضافة string إلى مجموعة عناصر ListBox أو CheckedListBox، على سبيل المثال :

```
myListBox.Items.Add( myListItem );
```

يمكن أن نكتب لإضافة سلسلة myListItem string إلى *myListBox* ListBox .

➢ لإضافة أغراض متعددة ، يمكننا إما استدعاء التابع Add عدة مرات أو استدعاء التابع AddRange لإضافة مصفوفة من الأغراض .

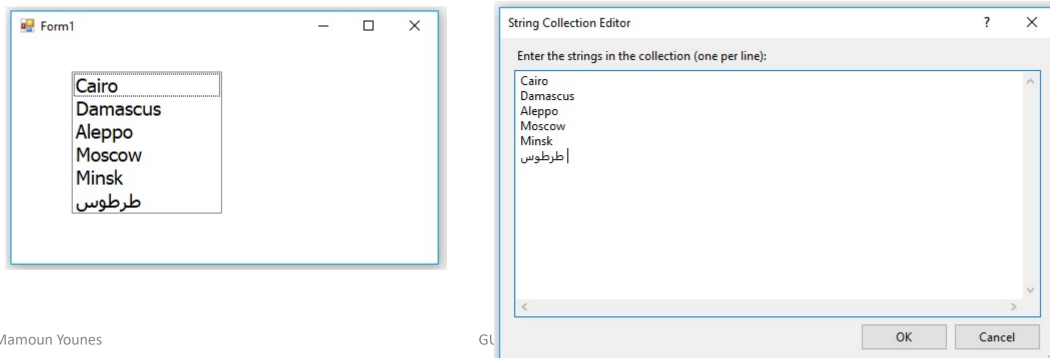
➢ يقوم كل من ListBox و CheckedListBox باستدعاء التابع ToString لتحديد Lable للغرض المرسل لإدخاله في القائمة ، هذا يسمح لنا بإضافة أغراض مختلفة إلى ListBox أو CheckedListBox التي يمكن إرجاعها لاحقًا من خلال الخصائص SelectedItem و SelectedItems .

➢ بدلاً من ذلك ، يمكنك إضافة عناصر إلى ListBoxes و CheckedListBoxes عن طريق فحص خاصية Items في خصائص Properties النافذة window .

مثال : صندوق القائمة ListBox

- **إضافة عناصر من التصميم :**

➢ نختار العنصر ListBox من الـ ToolBox ونسمه lbxCity ونختار الخاصية Collection (إي محرر النصوص) ونكتب فيه بعض أسماء المدن ، وعند تنفيذ البرنامج تظهر الواجهة التالية :



- **إضافة عناصر من الكود**

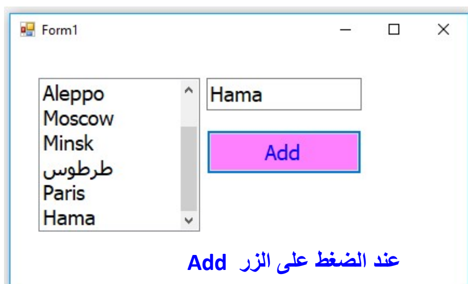
➢نضيف مربع نص textBox ونسميه txtCityName .

➢نضيف زر Button ونسميه btnAddCity ونكتب عليه كلمة Add ونكتب الكود التالي داخله :

```
private void btnAddCity_Click(object sender, EventArgs e)
        {
            string strCity = txtCityName.Text;
            lbxCity.Items.Add(strCity);
        }
```

➢ النص الموجود في textBox نضعه في متحول من نوع string .

➢ نضيف العنصر من textBox إلى ListBox بواسطة التابع Add ، وعند تنفيذ البرنامج تظهر الواجهة التالية :



• حذف عنصر من ListBox

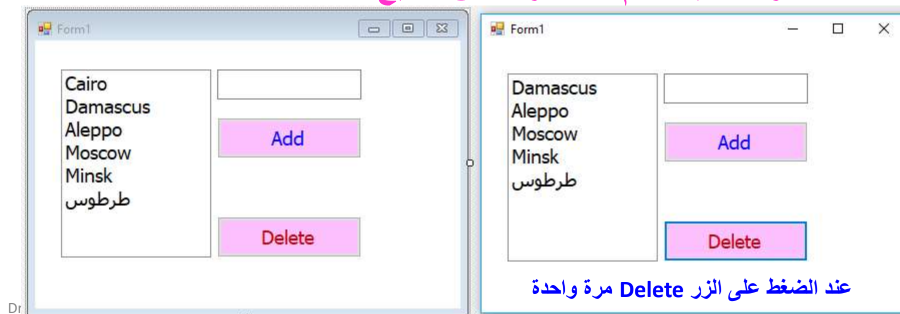
➢ نضيف زر Button ونسميه btnDel ونكتب عليه العبارة Delete ونكتب بداخل التابع الكود التالي :

```
private void btnDel_Click(object sender, EventArgs e)
        {
            lbxCity.Items.Remove("Cairo");

        }
```

➢ نستخدم التابع Remove لحذف أي عنصر من ListBox ، وعند تنفيذ البرنامج تظهر الواجهة التالية :

➢ **الطريقة الأولى لحذف عنصر : كتابة اسم العنصر ضمن التابع .**



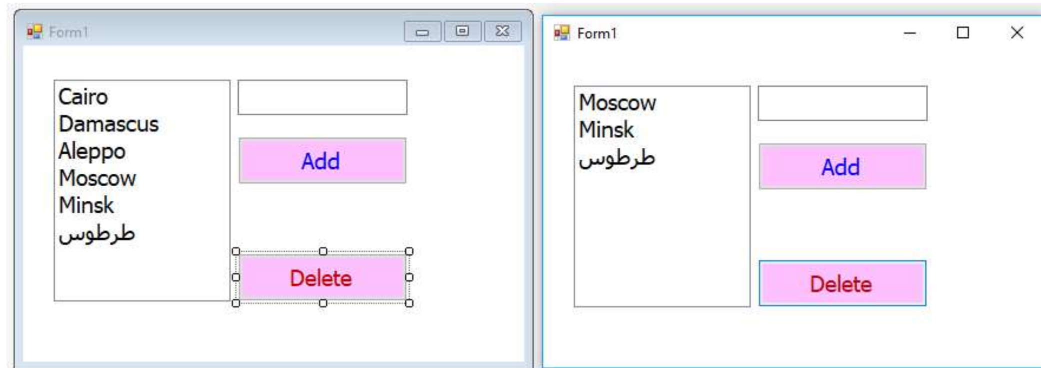
➢ **الطريقة الثانية لحذف عنصر**

نستخدم دليل العنصر (Index) في التابع RemoveAt(index) كما يلي :

```
private void btnDel_Click(object sender, EventArgs e)
        {
            lbxCity.Items.RemoveAt(0);
        }
```

➢ **هنا وضعنا 0 للدلالة على دليل العنصر الأول** ، وممكن وضع 1 او 2 أو إلى أي دليل موجود ضمن ListBox ، وعند حذف العنصر الأول يصبح العنصر الثاني هو العنصر الأول ، وهكذا إلى أن نحذف جميع العناصر بعد ذلك اذا تم حذف عنصر سوف يعطي لنا رسالة خطأ .

➢ **وعند تنفيذ الخرج تظهر الواجهة التالية :**



عند الضغط على الزر Delete ثلاث مرات

- لكي لا يعطي رسالة خطأ ويتوقف البرنامج ، يمكن كتابة الشرط التالي :

```
if(lbxCity.Items.Count > 0 )
  lbxCity.Items.RemoveAt(0);
```

- أي إذا كان عدد العناصر اكبر من الصفر نفذ الشرط وإلا توقف ، في هذه الحالة يتم حذف جميع العناصر .
- أذا أردنا حذف العنصر 2 نكتب : lbxCity.Items.RemoveAt(2) في هذه الحالة يتم حذف العنصر 2 ويصبح العنصر رقم 3 هو 2 ثم بالضغط على الزر Delete مرة أخرى يتم حذف العنصر وهكذا .
- ممكن جعل الشرط كما يلي :

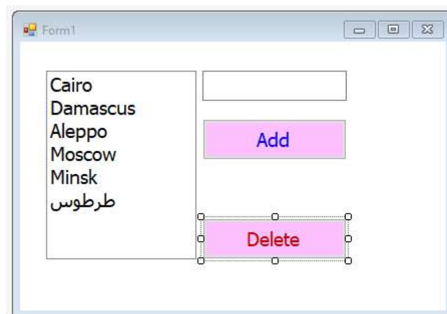
```
if(lbxCity.Items.Count > 2 )
  lbxCity.Items.RemoveAt(2);
```

- حذف جميع العناصر مرة واحدة من ListBox

➢ نستخدم التابع Clear() لحذف جميع العناصر من ListBox ، ونكتب الكود التالي :

```
private void btnDel_Click(object sender, EventArgs e)
        {
            lbxCity.Items.Clear();
        }
```

➢ وعند تنفيذ البرنامج تظهر الواجهة التالية :

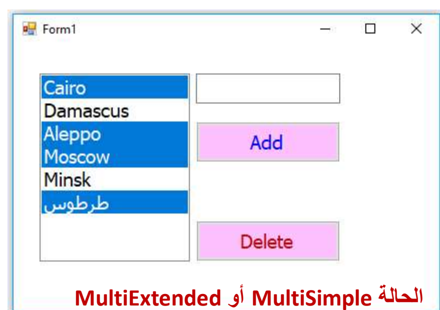


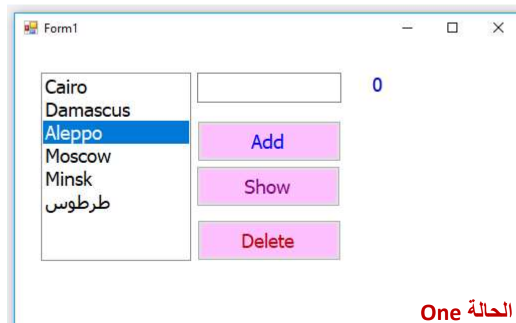


- يمكن تحديد أي عنصر ضمن Listbox باستخدام التابع ClearSelected() من خلال الخاصية SelectionMode ويوجد ثلاثة حالات كما يلي :

1. الحالة One وهي الحالة الافتراضية ، أي تحديد عنصر واحد فقط من Listbox .
2. الحالة MultiSimple أي تحديد أكثر من عنصر من Listbox بشكل سهل وبسيط .
3. الحالة MultiExtended أي تحديد أكثر من عنصر من Listbox مع الضغط على الزر shift أو Ctrl .

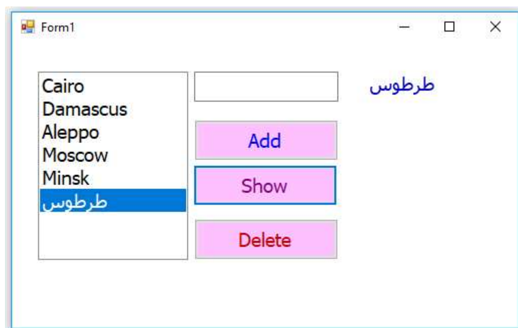
- عند تحديد أي عنصر أو عدد من العناصر يتم إزالة التحديد بواسطة الزر delete .

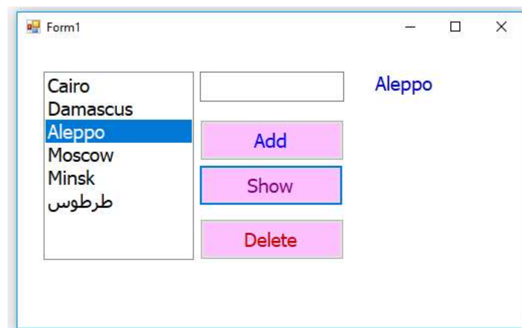




❑ إظهار العنصر الذي تم تحديده من ListBox

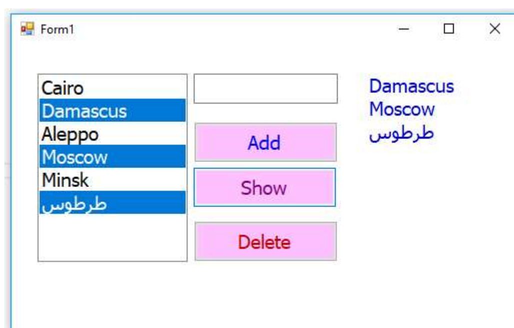
- ➢ إضافة زر Button إلى الواجهة ونسميه btnShow ونكتب عليه Show .
- ➢ إضافة لافتة Lable إلى الواجهة ونسميها lblMsg ونكتب القيمة 0 .
- ➢ الخاصية SelectionMode للعنصر ListBox في الحالة One .
- ➢ عند تنفيذ البرنامج تظهر الواجهة التالية :
- ➢ عند تحديد أي عنصر من ListBox و الضغط على الزر Show سوف يظهر العنصر على اللافتة Lable .

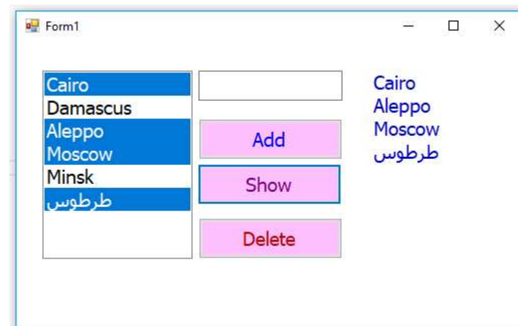




- **الخاصية SelectionMode للعنصر ListBox في الحالة MultiSimple أو الحالة MultiExtended .**

- ➢ عند تنفيذ البرنامج تظهر الواجهة التالية :
- ➢ عند تحديد عدد من العناصر من ListBox و الضغط على الزر Show سوف تظهر العناصر على اللافتة Lable .





البرنامج في حالة
One Selected

```
private void btnShow_Click(object sender, EventArgs e)
        {
             ////  One Selected
            lblMsg.Text = lbxCity.SelectedItem.ToString();
        }
```

البرنامج في حالة
Multi Selected

```
private void btnShow_Click(object sender, EventArgs e)
        {
            /////// Multi Selected
            lblMsg.Text = "";
            for (int x = 0; x < lbxCity.SelectedItems.Count; x += 1)
            {
                lblMsg.Text += lbxCity.SelectedItems[x] + "\n";
            }
        }
```


البرنامج الكامل للمثال : صندوق القائمة ListBox

```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication12
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnAddCity_Click(object sender, EventArgs e)
        {
            string strCity = txtCityName.Text;
            lbxCity.Items.Add(strCity);
        }
```

```
private void btnDel_Click(object sender, EventArgs e)
        {
            //lbxCity.Items.Remove("Cairo");
             //lbxCity.Items.RemoveAt(0);
             /*if(lbxCity.Items.Count > 0 )
            lbxCity.Items.RemoveAt(0);*/
            //  lbxCity.Items.Clear();
             lbxCity.ClearSelected();

        }
private void Form1_Load(object sender, EventArgs e)
        {

        }
private void lbxCity_SelectedIndexChanged(object sender, EventArgs e)
        {

        }
  private void txtCityName_TextChanged(object sender, EventArgs e)
        {

        }
```



```
private void btnShow_Click(object sender, EventArgs e)
        {
             //  One Selected
             lblMsg.Text = lbxCity.SelectedItem.ToString();
            /////// Multi Selected
            lblMsg.Text = "";
            for (int x = 0; x < lbxCity.SelectedItems.Count; x += 1)
            {
                lblMsg.Text += lbxCity.SelectedItems[x] + "\n";
            }
        }


}//end class

}// end namespace
```

صندوق قائمة الاختيار ChekedListBox

- **يُشتق عنصر التحكم CheckedListBox من ListBox** ويعرض CheckBox مع كل عنصر، ويمكن إضافة العناصر عبر طرائق Add و AddRange أو من خلال محرر String Collection. **وتسمح CheckedListBoxes بفحص عناصر متعددة** ، ولكن اختيار العناصر يكون أكثر تقييدًا.
- **القيم الوحيدة للخاصية SelectionMode هي None و One** ، أحدهما يسمح بتحديد مفرد ، بينما الآخر لا يسمح بلا تحديدات، نظرًا لأنه يجب تحديد عنصر ليتم التحقق منه ، **ويجب تعيين SelectionMode ليكون واحدًا** إذا كنا نرغب في السماح للمستخدمين بالتحقق من العناصر، وبالتالي ، يؤدي تبديل الخاصية SelectionMode بين One و None إلى التبديل بفعالية بين تمكين وتعطيل قدرة المستخدم على فحص عناصر القائمة. يُظهر الجدول(11) الخصائص العامة وطريقة وحدث CheckedListBoxes .

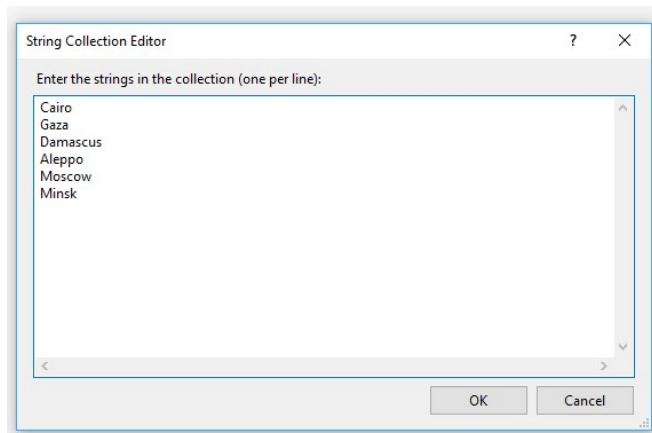
CheckedListBox properties, a method and an event	Description
Common Properties	*(All the ListBox properties, methods and events are inherited by CheckedListBox.)*
CheckedItems	Accessible only at runtime. Returns the collection of items that are checked as a CheckedListBox.CheckedItemCollection. This is distinct from the selected item, which is highlighted (but not necessarily checked). There can be at most one selected item at any given time.
CheckedIndices	Accessible only at runtime. Returns indices for all checked items as a CheckedListBox.CheckedIndexCollection.
CheckOnClick	When true and the user clicks an item, the item is both selected and checked or unchecked. By default, this property is false, which means that the user must select an item, then click it again to check or uncheck it.
SelectionMode	Determines whether items can be selected and checked. The possible values are One (the default; allows multiple checks to be placed) or None (does not allow any checks to be placed).
Common Method	
GetItemChecked	Takes an index and returns true if the corresponding item is checked.
Common Event (Event arguments ItemCheckEventArgs)	
ItemCheck	Generated when an item is checked or unchecked.
ItemCheckEventArgs Properties	
CurrentValue	Indicates whether the current item is checked or unchecked. Possible values are Checked, Unchecked and Indeterminate.
Index	Returns the zero-based index of the item that changed.
NewValue	Specifies the new state of the item.

الجدول(11) | CheckedListBox properties, a method and an event.

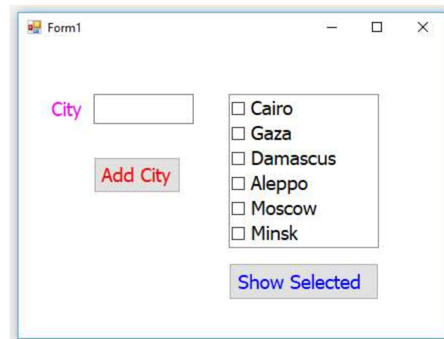
- يحدث الحدث ItemCheck عندما يقوم مستخدم بفحص أو إلغاء تحديد عنصر CheckedListBox ، وسائط الحدث CurrentValue و NewValue تقوم بإرجاع قيم CheckState للحالة الحالية والجديدة للعنصر ، على التوالي ، وتسمح لنا مقارنة هذه القيم بتحديد ما إذا كان عنصر CheckedListBox محددًا أو غير محدد.
- يحتفظ عنصر التحكم Checked-ListBox بخصائص SelectedItems و SelectedIndices (أنه يرث الصف ListBox) ومع ذلك ، فإنه يتضمن أيضًا خصائص CheckedItems و CheckedIndices، والتي تعرض معلومات حول العناصر والفهارس المحددة.
-

مثال : صندوق الاختيار CheckedListBox

- هذا العنصر يقوم بإنشاء قائمة من صناديق الاختيار ضمن صندوق أو نافذة ، ومن الخاصية Item نختار Collection فتفتح لنا نافذة نكتب بها أسماء بعض المدن وهي تعتبر صناديق اختيار.



- نضغط على الزر OK فتتشكل لنا القائمة من صناديق الاختيار ، وعند تنفيذ البرنامج تظهر لنا النافذة التالية :



- نلاحظ ظهور أسماء المدن كقائمة من صناديق الاختيار ، وعند الضغط على أي واحدة لا يتم وضع علامة على صندوق الاختيار، أولاً يجب تعليمها ثم النقر عليها لوضع علامة ضمن صندوق الاختيار .
- نختار الخاصية CheckOnClick ونغيرها من حالة false إلى حالة true ، في هذه الحالة عند الضغط على زر الاختيار يتم وضع علامة ضمن صندوق الاختيار مباشرة .

- ننشئ زر Button ونسميه btnShow ونكتب عليه Show Selected ، ونكتب داخل التابع ما يلي :

```
private void btnShow_Click(object sender, EventArgs e)
 {
     for (int x = 0; x < checkedListBox1.CheckedItems.Count; x++)
     {

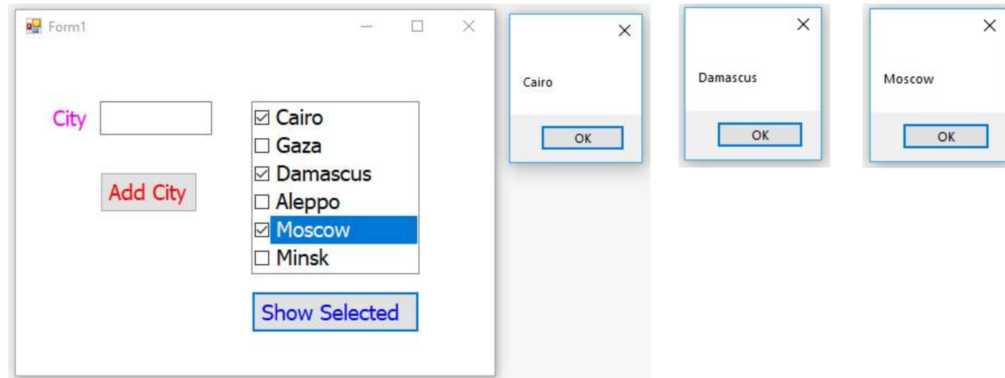
      MessageBox.Show(checkedListBox1.CheckedItems[x].ToString());
     }
 }
```

- تقوم الحلقة for بإظهار أسماء المدن بشكل متكرر على صندوق الحوار MessageBox.Show حسب عدد مرات تكرار الحلقة for ، حيث :

checkedListBox1.CheckedItems.Count

عدد العناصر التي تم اختيارها

العناصر التي تم اختيارها من قائمة صناديق الاختيار



- إذا أردنا أن نضيف إلى القائمة عناصر اختيار أخرى ، نقوم بإنشاء زر Button ونسميه button1 ونكتب عليه Add City ، وننشئ لافته Label ونسميها City ، وننشئ مربع نص textBox1 ، ونكتب في الزر Add City العبارة البرمجية التالية :

```
private void button1_Click(object sender, EventArgs e)
        {
            checkedListBox1.Items. Add(textBox1.Text) ;
        }
```

حيث :

Add(textBox1.Text) أي إضافة النص الذي يحتويه مربع النص textBox1 إلى قائمة صناديق الاختيار ، فيزداد عددها .

- نكتب اسم المدينة في مربع النص ونضغط على الزر Add City فيتم إضافة اسم المدينة إلى القائمة و هكذا ..
- يوضح البرنامج التالي جميع الحالات السابقة .

البرنامج

```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication10
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnShow_Click(object sender, EventArgs e)
        {
            for (int x = 0; x < checkedListBox1.CheckedItems.Count; x++)
            {
                MessageBox.Show(checkedListBox1.CheckedItems[x].ToString());
            }
        }
```



```
private void checkedListBox1_SelectedIndexChanged(object sender, EventArgs e)
        {

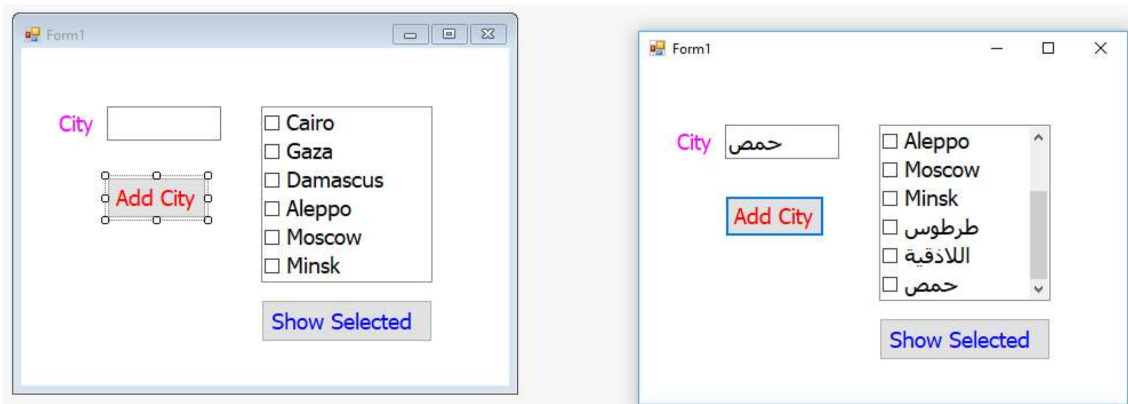
        }

        private void button1_Click(object sender, EventArgs e)
        {
            checkedListBox1.Items.Add(textBox1.Text);
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }
 }//end class

}// end namespace
```



أزرار الاختيار RadioButton

- تشبه أزرار الاختيار (التي يتم تعريفها من خلال الصف RadioButton) صناديق الاختيار CheckBoxes بأنها تحتوي أيضًا على حالتين - محددة selected وغير محددة not selected ومع ذلك ، تظهر RadioButtons عادة كمجموعة group، حيث يمكن تحديد RadioButton واحد فقط في كل مرة.
- اختيار زر واحد RadioButton في المجموعة يجبر الآخرين على أن يتم إلغاء تحديدهم، لذلك ، يتم استخدام أزرار الاختيار RadioButtons لتمثيل مجموعة من الخيارات الحصرية المتبادلة (لا يمكن تحديد خيارات متعددة في نفس الوقت).
- **جميع أزرار RadioButtons يجب أن تضاف إلى حاوية مثل : GroupBoxes أو Panels .**
- ويبين الجدول (11)الخصائص العامة والحدث العام للصف RadioButton

RadioButton properties and an event	Description
Common Properties	
Checked	Indicates whether the RadioButton is checked.
Text	Specifies the RadioButton's text.
Common Event	
CheckedChanged	Generated every time the RadioButton is checked or unchecked. When you double click a RadioButton control in design view, an empty event handler for this event is generated.

الجدول(11) | RadioButton properties and an event.

Checked : يشير إلى ما إذا كان RadioButton محددًا أم لا
Text : يحدد اسم لزر الاختيار .

صناديق المجموعة وللوحات
GroupBoxs and Panels

- **تُستخدم GroupBoxes وPanels في واجهة المستخدم المرئية** ، ويتم استخدام GroupBoxes و Panelsعادةً لتجميع عدة عناصر تحكم لوظائف مشابهة أو عناصر تحكم متعددة مرتبطة بـ GUI ، **وتتحرك كافة عناصر التحكم في GroupBox أو Panel معاً عند نقل GroupBox أو Panel**، علاوة على ذلك ، يمكن أيضًا استخدام GroupBoxes وPanels لإظهار أو إخفاء مجموعة من عناصر التحكم في الوقت نفسه.
- عند تعديل خاصية Visible للحاوية ، فإنها تعمل على تبديل مستوى رؤية جميع عناصر التحكم بداخلها .
- **الفرق الأساسي بين عنصري التحكم هذين** هو أن GroupBoxes يمكنها عرض التسمية التوضيحية (أي نص) ولا تتضمن أشرطة التمرير ، بينما يمكن أن تتضمن اللوحات أشرطة التمرير ولا تتضمن تسمية توضيحية.

• يكون لدى **GroupBoxes** حدود صغيرة بشكل افتراضي؛ ويمكن تعيين لوحات بحيث يكون لها أيضًا حدود بتغيير خاصية BorderStyle الخاصة بها. ويوضح الجدول(12) الخصائص العامة لـ GroupBoxes والجدول(13) الخصائص العامة لـ Panel.

GroupBox properties	Description
Controls	The set of controls that the GroupBox contains.
Text	Specifies the caption text displayed at the top of the GroupBox.

الجدول(12) | GroupBox properties.

Panel properties	Description
AutoScroll	Indicates whether scrollbars appear when the Panel is too small to display all of its controls. The default value is false.
BorderStyle	Sets the border of the Panel. The default value is None; other options are Fixed3D and FixedSingle.
Controls	The set of controls that the Panel contains.

الجدول(13) | Panel properties.

مثال : انشاء لافتة وأربعة أزرار اختيار من أجل كتابة نص في اللافتة ثم استخدام أزرار الاختيار لجعل الخط Plain ,Italic , Bold , Italic/Bold

البرنامج

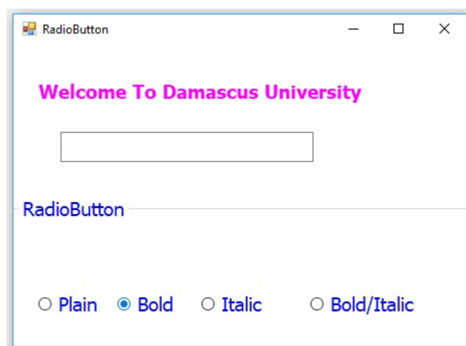
```
using System;
using System.Drawing;
using System.Windows.Forms;
namespace WindowsFormsApplication7
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

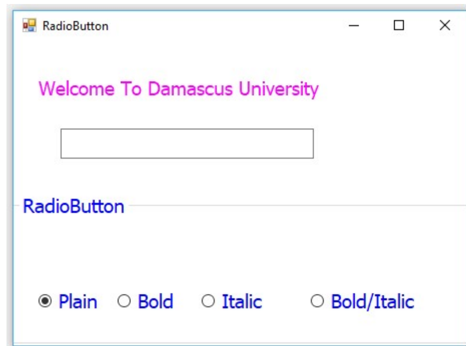
        private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {

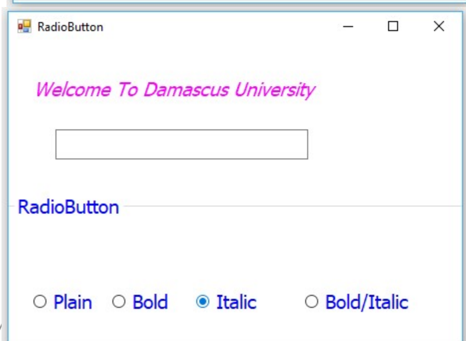
        }
```

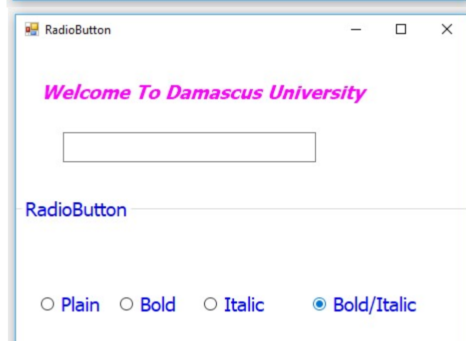
```
private void radioButton2_CheckedChanged(object sender, EventArgs e)
        {
            outputlabel1.Font = new Font(outputlabel1.Font,
                                   outputlabel1.Font.Style ^ FontStyle.Italic);
        }
        private void radioButton3_CheckedChanged(object sender, EventArgs e)
        {
            outputlabel1.Font = new Font(outputlabel1.Font,
                                   outputlabel1.Font.Style ^ FontStyle.Bold);
        }
        private void radioButton4_CheckedChanged(object sender, EventArgs e)
        {
              outputlabel1.Font = new Font(outputlabel1.Font,
                      outputlabel1.Font.Style ^( FontStyle.Bold | FontStyle.Italic));
        }

  }//end class
}// end namespace
```



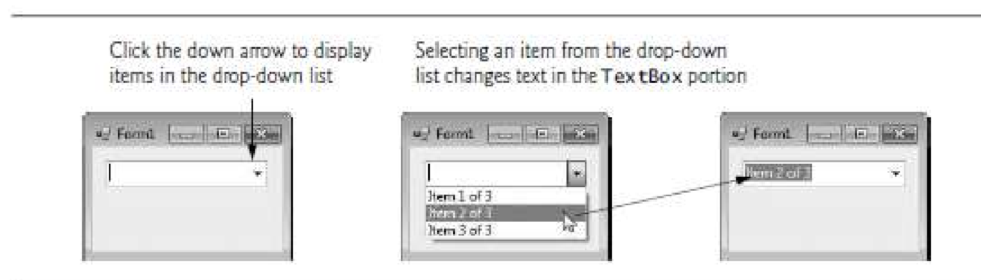




صندوق اختيار متاح
ComboBox

- **يتمتع عنصر التحكم ComboBox بميزات TextBox مع قائمة منسدلة drop-down list الذي يحتوي على قائمة يمكن تحديد قيمة منها.**
- يظهر ComboBox عادةً كـ TextBox مع سهم للأسفل إلى اليمين.
- افتراضيًا ، يمكن للمستخدم إدخال نص في مربع النص أو النقر على السهم لعرض قائمة بالعناصر المحددة مسبقًا ، وإذا اختار المستخدم عنصرًا من هذه القائمة ، فسيتم عرض هذا العنصر في مربع النص.
- إذا كانت القائمة تحتوي على عناصر أكثر مما يمكن عرضه في القائمة المنسدلة ، فسيظهر شريط تمرير *scrollbar*.

- يتم تعيين الحد الأقصى لعدد العناصر التي يمكن عرض قائمة منسدلة في وقت واحد بواسطة الخاصية MaxDrop-DownItems ، و يعرض الشكل (1) نموذج ComboBox في ثلاث حالات مختلفة.



الشكل(1) | ComboBox demonstration.

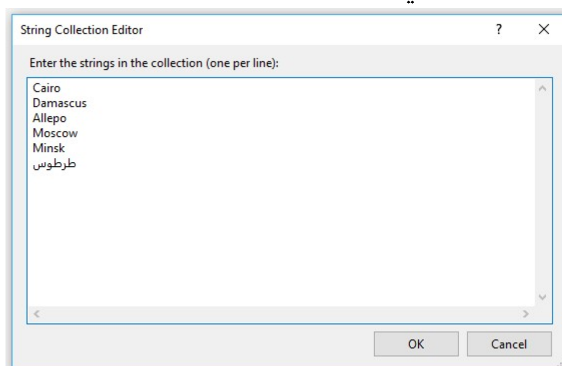
- كما هو الحال مع عنصر التحكم ListBox، يمكننا إضافة أغراض إلى مجموعة العناصر برمجياً ، باستخدام التوابع Add و AddRange ، أو بشكل مرئي ، مع محرر مجموعة النصوص String Collection Editor ن ويعرض الجدول(13) الخصائص العامة والحدث العام للصف ComboBox .

ComboBox properties and an event	Description
Common Properties	
DropDownStyle	Determines the type of ComboBox. Value Simple means that the text portion is editable and the list portion is always visible. Value DropDown (the default) means that the text portion is editable but the user must click an arrow button to see the list portion. Value DropDownList means that the text portion is not editable and the user must click the arrow button to see the list portion.
Items	The collection of items in the ComboBox control.

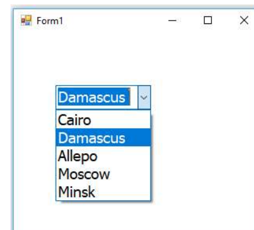
الجدول(13) | ComboBox properties and an event. (Part 1 of 2.)

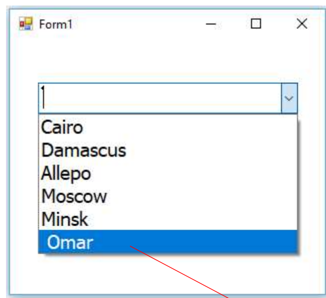
مثال : صندوق اختيار متاح ComboBox

- تجميع عدد من العناصر على شكل قائمة مخفية ، وعند الضغط عليها تظهر جميع العناصر التي بداخلها ، ويمكن اختيار أي عنصر منها بالضغط عليه .
- ننشئ ComboBox ونسميه cbxCity ، نختار الخاصية Items ونختار Collection فتظهر لنا نافذة (محرر نصوص) ونكتب العناصر كما يلي :



- نضغط على Ok يتم حفظ العناصر بداخلها .
- ننفذ البرنامج نحصل على الواجهة التالية .





- الطريقة الثانية لإدخال العناصر إلى ComboBox كما يلي :
  - انشاء زر اسمه btnAddItem ونكتب عليه Add Item .
  - نكتب داخل التابع للزر البرنامج التالي لإدخال العناصر عن طريق الضغط على الزر .
  - كلما نضغط يتم ادخال عنصر ، نضغط مرتين يتم ادخال نفس العنصر مرتين و هكذا .

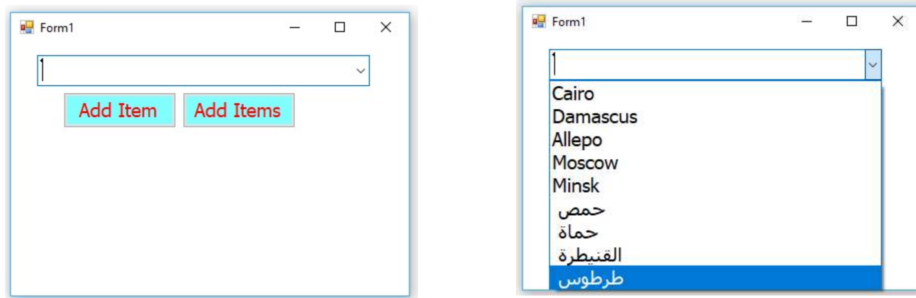
```
private void btnAddItem_Click(object sender, EventArgs e)
        {
            cbxCity.Items.Add(" Omar");
        }
```

ادخل العنصر Omar إلى عناصر الـ ComboBox

- الطريقة الثالثة لإدخال العناصر إلى الـ ComboBox كما يلي :
  - انشاء زر ثاني ونسميه btnAddItems ونكتب عليه Add Items .
  - نكتب داخل التابع للزر btnAddItems البرنامج التالي :

```
private void btnAddItems_Click(object sender, EventArgs e)
        {
            String []str = { "طرطوس", "القنيطرة" , "حماة", "حمص"  } ;
             cbxCity.Items.AddRange(str);
        }
```

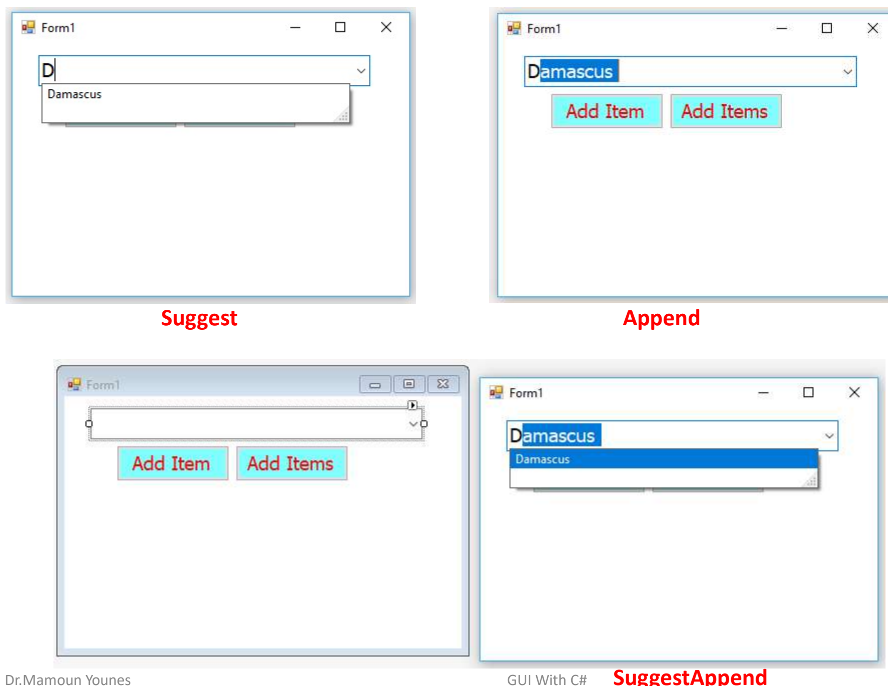
- تم انشاء مصفوفة محرفية .
- العبارة cbxCity.Items.AddRange إضافة جميع عناصر المصفوفة إلى الـ ComboBox ، عند الضغط على الزر Add Items .
- عند تنفيذ البرنامج تظهر الوجهة التالية .



- إذا أردنا البحث عن كلمة موجودة في الـ ComboBox نختار الخاصتين :

➢ الخاصية الأولى AutoCompleteMod وفيها : Suggest , Append , SuggestAppent ، وتعني اقتراح الكلمة ، تكملة الكلمة المقترحة ، اقتراح بالإضافة إلى تكملة الكلمة المقترحة .

➢ الخاصية الثانية AutoCompleteSource نختار منها الـ ListItems ، أي اقتراح الكلمة هي من القائمة الموجودة في الـ ComboBox وهي مصدر الكلمات .



Suggest

Append

SuggestAppend

البرنامج

```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication11
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnAddItem_Click(object sender, EventArgs e)
        {
            cbxCity.Items.Add(" Omar");
        }
```



```
private void button1_Click(object sender, EventArgs e)
        {

        }

        private void btnAddItems_Click(object sender, EventArgs e)
        {
          String []str = { "طرطوس", "القنيطرة" , "حماة", "حمص   "} ;

           cbxCity.Items.AddRange(str);
        }

        private void cbxCity_SelectedIndexChanged(object sender, EventArgs e)
        {

        }
    }//end class
}// end namespace
```

إظهار وإخفاء العناصر
Visible and Hide

- الخاصية **Visible** عند جعلها بحالة true تقوم بإظهار العنصر .
- الخاصية **Hide** عند جعلها بحالة false تقوم بإخفاء العنصر .
- **إظهار العناصر وإخفاءها نستخدم الخاصيتين Visible و Hide** ، فالإخفاء لا يعني الحرمان من التعامل معه ، وبالتالي الإخفاء لا يعني الغاء العنصر .
- البرنامج التالي يوضح كيف يتم الاستفادة من الإخفاء والإظهار .

مثال : انشاء ثلاثة أزرار Button ، وثلاثة مربعات نص textBox ، وصندوق اختار CheckBox ، وصندوق comboBox .

- نضغط على مربع النص textBox1 ونختار الخاصية Hide ، ونكتب في مربع النص العبارة text1 ، فيختفي مربع الحوار مع العبارة عند تنفيذ البرنامج .
- نكتب في التابع للزر button1 العبارة التالية :

```
MessageBox.Show(textBox1.Text);
```

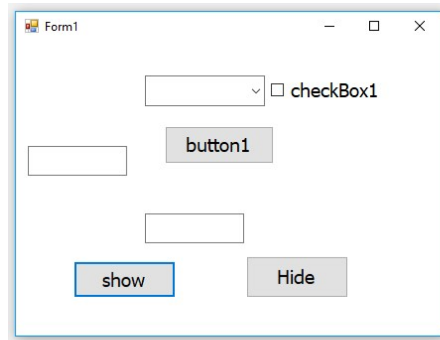
- عند تنفيذ البرنامج تظهر العبارة المخفية في صدوق الحوار .
- نكتب على الزر Button2 العبارة Visible ، بينما نكتب على الزر Button3 العبارة false .
- نكتب في التابع للزر button2 العبارة التالية :

```
txtTest.Visible = true;
```

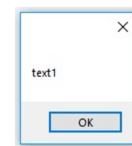
- نكتب في التابع للزر button3 العبارة التالية :

```
txtTest.Visible = false;
```

- ننفذ البرنامج فتظهر لنا الواجهة التالية :

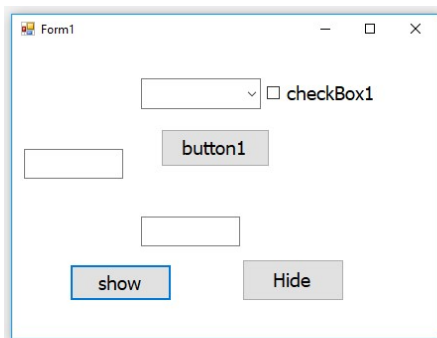


- عند الضغط على الزر Button1
فيظهر صندوق الحوار التالي :

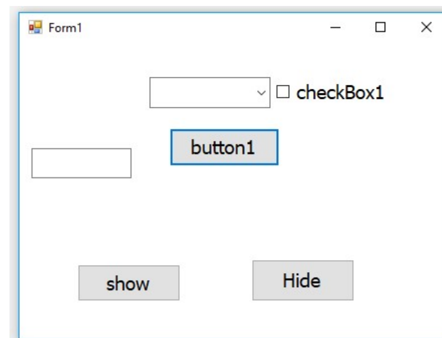


أي أن النص ظهر في صندوق الحوار رغم أن مربع النص مخفي .

عند الضغط على الزر Hide سوف يختفي مربع النص textBox2 ، وعند الضغط على الزر Visible سوف يظهر مرة أخرى ، وهذا نكرر الضغط بين الزرين .



عند الضغط على الزر Visible



عند الضغط على الزر Hide

البرنامج الكامل

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication8 {
 public partial class Form1 : Form {
  public Form1()
        {
            InitializeComponent();
        }
private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show(textBox1.Text);
        }
private void btnVisible_Click(object sender, EventArgs e)
        {
            txtTest.Visible = true;
        }
private void btnHide_Click(object sender, EventArgs e)
        {
            txtTest.Visible = false;
        }
```



```
private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }
 private void txtTest_TextChanged(object sender, EventArgs e)
        {

        }
private void textBox3_TextChanged(object sender, EventArgs e)
        {

        }
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {

        }
private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {

        }
    }//end class
}// end namespace
```

التحكم بموقع العناصر
Control the location of items

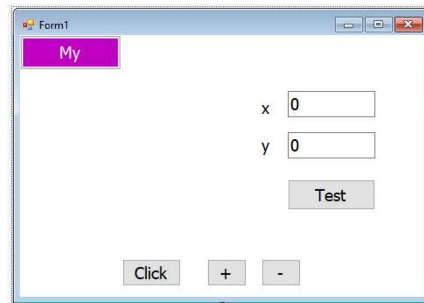
- يمكن ألتحكم بموقع أي عنصر في ألواجهة ألرسومية وألتحكم بحركته ضمن ألواجهة المرئية ، ومن الجدير بالذكر أن مبدأ الإحداثيات للواجهة المرئية هو الزاوية اليسارية العليا ، مع العلم بأن اتجاه محوري الرسم موضحان في الشكل التالي :

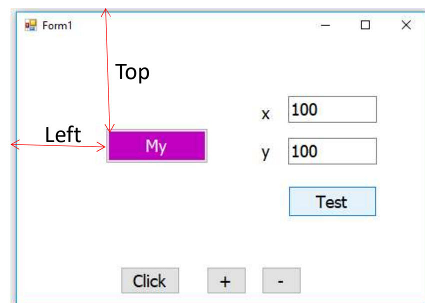


- يوجد عدد من الطرائق منها :

1. بواسطة تغيير إحداثيتي العنصر عن طريق x و y ، ثم الضغط على الزر Test يتم تحريك الزر My إلى الموقع المحدد (حيث : Left و Top تدل على x و y) كما يلي :

```
private void btnTest_Click(object sender, EventArgs e)
        {
            int x = int.Parse(textBox1.Text);
            int y = int.Parse(textBox2.Text);
            btnMy.Left = x;
            btnMy.Top = y;
        }
```



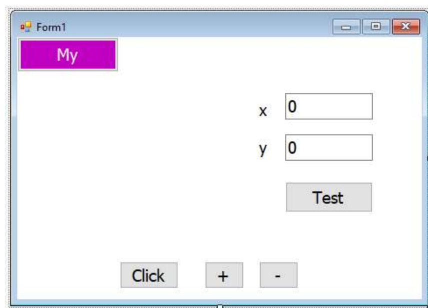


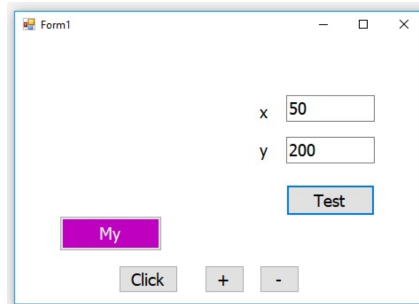
2. الطريقة الثانية استخدام الخاصية Location واستخدام الصف Point من أجل تمرير الاحداثيات x و y ، كما يلي :

```
private void btnTest_Click(object sender, EventArgs e)
        {
            btnMy.Location = new Point(x, y);

        }
```

نلاحظ أن هذه الطريقة أسهل .





3. الطريقة الثالثة استخدام الخاصية Location واستخدام الصف Point من أجل تمرير الاحداثيات 1 + btnMy.Left و btnMy.Top+1 ، وذلك بالضغط على الزر + يتم تحريك الزر My باتجاه المحور x ، وعند الضغط على الزر - يتم تحريك الزر My باتجاه المحور y كما يلي :

```
private void button1_Click(object sender, EventArgs e)
        {
            btnMy.Location = new Point(btnMy.Left + 1, btnMy.Top+1);

        }

        private void button2_Click(object sender, EventArgs e)
        {
            btnMy.Location = new Point(btnMy.Left - 1, btnMy.Top - 1);

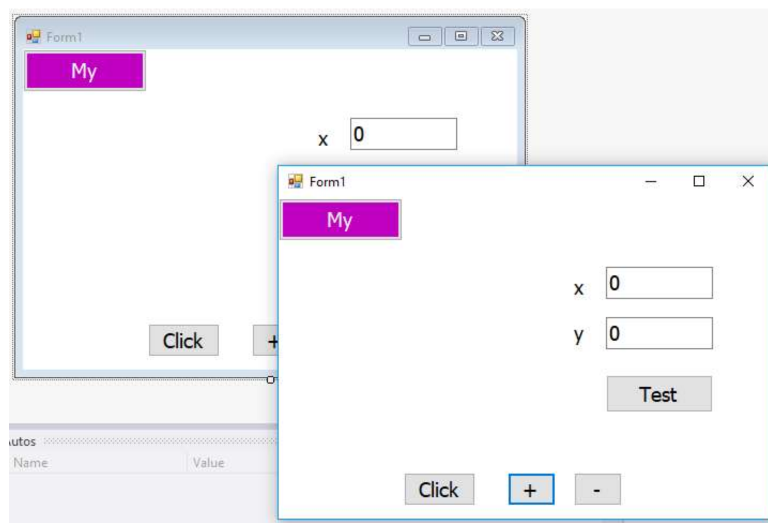

        }
```

4. الطريقة الرابعة هو تحريك النافذة Form مع العناصر جميعها الخاصية this.Left لتحريك النافذة بتجاه المحور x مع ثبات المحور y ، أما الخاصية this.Top لتحريك النافذة بتجاه المحور y مع ثبات المحور x ، كما يلي :

```
private void button1_Click(object sender, EventArgs e)
        {
             this.Left += 1;
             this.Top += 1;
        }

        private void button2_Click(object sender, EventArgs e)
        {
             this.Left -= 1;
             this.Top -= 1;

        }
```

البرنامج الكامل

```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication9
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnTest_Click(object sender, EventArgs e)
        {
            int x = int.Parse(textBox1.Text);
            int y = int.Parse(textBox2.Text);
             /*btnMy.Left = x;
            btnMy.Top = y; */
            btnMy.Location = new Point(x, y);
        }
```



```
private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }
private void textBox2_TextChanged(object sender, EventArgs e)
        {

        }
private void button1_Click(object sender, EventArgs e)
        {
            // btnMy.Location = new Point(btnMy.Left + 1, btnMy.Top+1);
             this.Left += 1;
             this.Top += 1;
        }

private void button2_Click(object sender, EventArgs e)
        {
            // btnMy.Location = new Point(btnMy.Left - 1, btnMy.Top - 1);
             this.Left -= 1;
             this.Top -= 1;

        }
```

```
private void btnMy_Click(object sender, EventArgs e)
        {

        }

private void label1_Click(object sender, EventArgs e)
        {

        }

private void label2_Click(object sender, EventArgs e)
        {

        }
    }//end class
}// end namespace
```


صناديق الصورة
PictureBox

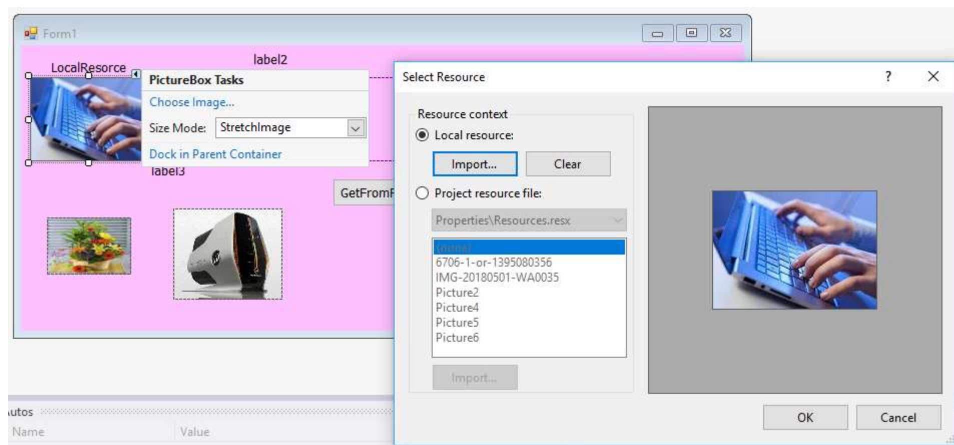
- يعرض ألعنصر PictureBox صورة. ، ويمكن أن تكون ألصورة واحدة من عدة تنسيقات ، مثل الصورة النقطية bitmap وPNG (Portable Network Graphics) وGIF (Graphics Interchange Format) و JPEG .
- تحدد خاصية ImageBox Image الصورة المعروضة ، وتشير الخاصية SizeMode إلى كيفية عرض الصورة (Normal, StretchImage, Autosize, CenterImage أو Zoom) ويوضح الشكل الجدول(15) خصائص PictureBox العامة والحدث العام .
-

PictureBox properties and an event	Description
Common Properties	
Image	Sets the image to display in the PictureBox.
SizeMode	Enumeration that controls image sizing and positioning. Values are Normal (default), StretchImage, AutoSize, CenterImage, and Zoom. Normal places the image in the PictureBox's top-left corner, and CenterImage puts the image in the middle. These two options truncate the image if it's too large. StretchImage resizes the image to fit in the PictureBox. AutoSize resizes the PictureBox to hold the image. Zoom resizes the image to to fit the PictureBox but maintains the original aspect ratio.
Common Event	
Click	Occurs when the user clicks a control. When you double click this control in the designer, an event handler is generated for this event.

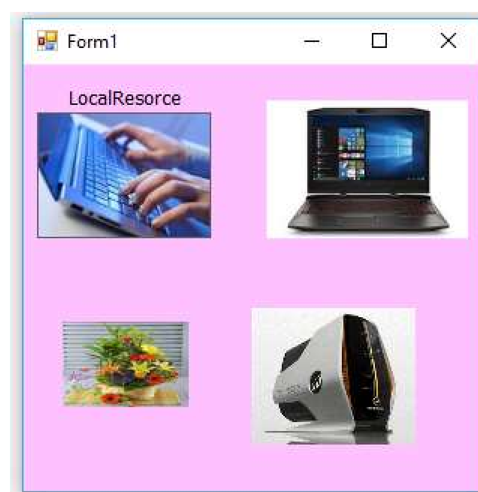
الجدول(15) | PictureBox properties and an event.

مثال : صندوق الصور PictureBox

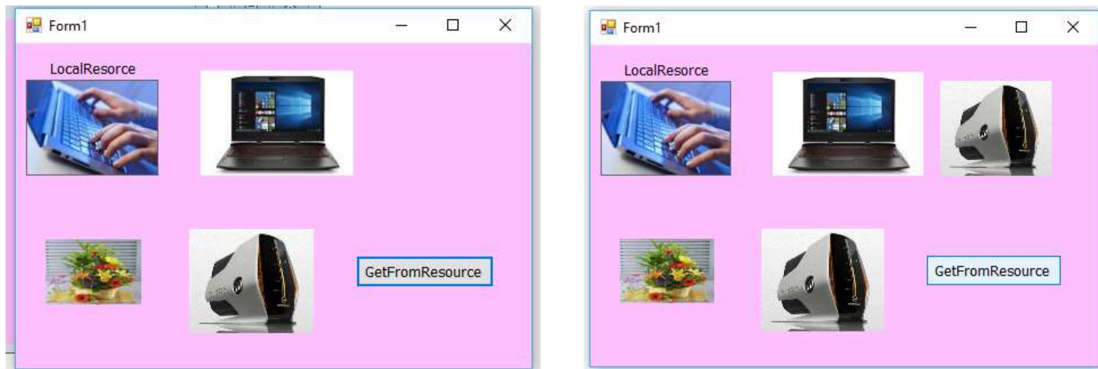
- إضافة عنصر أول PictureBox ونسميه Pic1 ، من الخاصية Image نختار الحالة StretchImage والتي تعني جعل حجم الصورة يناسب حجم PictureBox ، نختار الخاصية Choose Image ، تظهر لنا نافذة ، لدينا خيارين Local Resource أي لا نستطيع تكرار الصورة بعد جلبها ، أما الخيار الثاني Project Resource File الذي يعني يمكن تكرار الصورة بعد جلبها لأنها محفوظة في الملف ، ونجلب صورة بشكل Local Resource .
- إضافة عنصر ثاني PictureBox ونسميه Pic2 ونجلب صورة بشكل Project Resource File .
- إضافة عنصر ثالث ورابع PictureBox ونسميه Pic3,Pic4 ونجلب صورتين بشكل Project Resource File .



- عند تنفيذ البرنامج نحصل على الواجهة التالية :



• نضيف زر Button ونسميه btnGetFormResouce ونكتب عليه GetFormResouce ، بحيث يتم جلب صورة من Select Resource أي الواجهة التي حصلنا عليها من أجل جلب الصور .



البرنامج الكامل للمثال : صندوق الصور PictureBox

البرنامج

```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication13
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnGetFromResource_Click(object sender, EventArgs e)
        {
             pic5.Image = Properties.Resources.Picture4;
            //pic5.Image = Properties.Resources.IMG_20180501_WA0035;
        }
```

```
private void pic1_Click(object sender, EventArgs e)
        {

        }
private void pic2_Click(object sender, EventArgs e)
        {

        }
 private void pict3_Click(object sender, EventArgs e)
        {

        }
private void pic4_Click(object sender, EventArgs e)
        {

        }
 private void pic5_Click(object sender, EventArgs e)
        {

        }
    }//end class
}// end namespace
```


أحداث الفأرة
Mouth Event Handling

- يشرح هذا القسم كيفية التعامل مع أحداث الفأرة ، مثل النقر clicks والتحريك moves ، التي يتم إنشاؤها عند تفاعل المستخدم مع عنصر تحكم عبر الفأرة.
- يمكن معالجة أحداث الفأرة لأي عنصر تحكم مشتق من الصف System.Windows.Forms.Control .
- بالنسبة لمعظم أحداث الفأرة، يتم تمرير معلومات حول الحدث إلى تابع معالجة الحدث من خلال غرض من الصف MouseEventArgs، والنائب delegate المستخدم لإنشاء معالجات أحاث الفأرة هو MouseEventHandler .
- يتطلب كل تابع معالج حدث الفأرة لهذه الأحداث غرض Object ، وغرض من الصف MouseEventArgs كوسيط .

- يحتوي الصف MouseEventArgs على معلومات متعلقة بحدث الفأرة ، مثل إحداثيات x-and y للمؤشر الخاص بالفارة ، وضغط زر الفأرة(Right, Left or Middle) وعدد مرات النقر بالفأرة.
- إن إحداثيات x و y لغرض من Mouse- EventArgs هي نسبة إلى عنصر التحكم الذي أنشأ الحدث — على سبيل المثال ، النقطة (0،0) تمثل الزاوية اليسارية العليا لعنصر التحكم الذي وقع فيه حدث الفأرة.
- ويوضح الجدول(16) العديد من أحداث الفأرة العامة ووسائط الحدث .
- يستخدم البرنامج أحداث الفأرة للرسم على النافذة Form ، وعندما يقوم المستخدم بسحب الفأرة (أي تحريك الماوس أثناء الضغط على زر الماوس) ، تظهر دوائر صغيرة في Form في الموضع الذي يحدث فيه كل حدث ماوس أثناء عملية السحب.

Mouse events and event arguments	
Mouse Events with Event Argument of Type `EventArgs`	
`MouseEnter`	Mouse cursor enters the control's boundaries.
`MouseHover`	Mouse cursor hovers within the control's boundaries.
`MouseLeave`	Mouse cursor leaves the control's boundaries.
Mouse Events with Event Argument of Type `MouseEventArgs`	
`MouseDown`	Mouse button is pressed while the mouse cursor is within a control's boundaries.
`MouseMove`	Mouse cursor is moved while in the control's boundaries.
`MouseUp`	Mouse button is released when the cursor is over the control's boundaries.
`MouseWheel`	Mouse wheel is moved while the control has the focus.
Class `MouseEventArgs` *Properties*	
`Button`	Specifies which mouse button was pressed (`Left`, `Right`, `Middle` or `None`).
`Clicks`	The number of times that the mouse button was clicked.
`X`	The *x*-coordinate within the control where the event occurred.
`Y`	The *y*-coordinate within the control where the event occurred.

الجدول(16) | Mouse events and event arguments.

البرنامج الكامل : أحداث الفأرة Mouse events

البرنامج

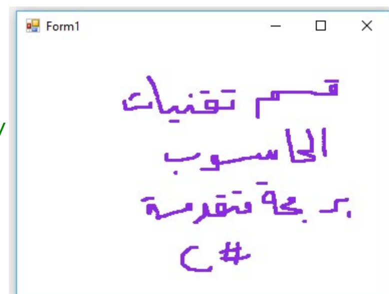
```
using System;
using System.Drawing;
using System.Windows.Forms;
namespace WindowsFormsApplication14
{
public partial class Form1 : Form
    {
        bool shouldPaint = false; // 8 //determines whether to paint
        public Form1()
        {
            InitializeComponent();
        }
private void Form1_Load(object sender, EventArgs e)
        {

        }
```



```
private void Form1_MouseDown(object sender, MouseEventArgs e)
        {
            shouldPaint = true; //  18
        }
private void Form1_MouseUp(object sender, MouseEventArgs e) //
        {
            shouldPaint = false; // 22
        }

private void Form1_MouseMove(object sender, MouseEventArgs e)
        {
            if (shouldPaint) //25 //check if mouse button is being pressed
            {
                // draw a circle where the mouse pointer is present
                using (Graphics graphics = CreateGraphics()) // 28
                {
                    graphics.FillEllipse(
                    new SolidBrush(Color.BlueViolet), e.X, e.Y, 4, 4);
                } // end using; calls graphics.Dispose()
            } // end if
        } // // // // // // 35
    }//end class
}// end namespace
```



- في السطر 18 ، يقوم البرنامج بتعريف متغير shouldPaint ، والذي يحدد ما إذا كان سيتم الرسم على الشاشة Form ، ونريد أن نرسم عندما يتم الضغط على زر الفأرة أي الضغط باستمرار ، وبالتالي عندما يقوم المستخدم بالنقر أو الضغط على زر الفأرة ، فإن النظام يقوم بإنشاء حدث MouseDown، ويجب أن يقوم معالج الأحداث (الأسطر 16–19) بتحديد الصلاحيات على true.
- عندما يقوم المستخدم بتحرير زر الفارة ، يقوم النظام بإنشاء حدث MouseUp، ويجب ان يكون المتغير shouldPaintعلى false في معالج PainterForm_MouseUp الأسطر(20–23) ثم يتوقف البرنامج عن الرسم ، بخلاف أحداث MouseMove، التي تحدث بشكل مستمر أثناء تحريك المستخدم للفارة ، ويقوم النظام بإنشاء حدث MouseDown فقط عند الضغط على زر الفأرة أولاً ويقوم بإنشاء حدث MouseUp فقط عند تحرير زر الفأرة.

- كلما تحركت الفأرة عبر عنصر تحكم ، يحدث الحدث MouseMove لعنصر التحكم.
- يرسم البرنامج فقط داخل معالج الأحداث PainterForm_MouseMoveخطوط (24–48) ، إذا كان المتغير shouldPaint في حالة true (أي ، يتم الضغط على زر الماوس).

الكلمة المحجوزة using في السطر 28 هو استدعاء الـForm الذي يرث التابع CreateGraphics لإنشاء غرض من الصف Graphics الذي يسمح للبرنامج بالرسم على Form.

يوفر الصف Graphics طرائق لرسم أشكال متعددة ، على سبيل المثال ، تستخدم السطور (24-35)التابع FillEllipse لرسم دائرة. الوسيط الأول في التابع FillEllipse هو غرض من الصف SolidBrush، والذي يحدد اللون الذي سيملأ الشكل. يتم توفير اللون كوسيط لباني الصف SolidBrush .

- يحتوي الصف Color على ثوابت لون محددة مسبقا - اخترنا Color.BlueViolet.
- يرسم التابع FillEllipse شكل بيضاوي في مستطيل حيث الإحداثيات x و y هي احداثيات الزاوية اليسارية العليا ، وعرضه W وارتفاعه H (الوسائط الأربع الأخيرة للتابع .
- تمثل الإحداثيات x و y موقع حدث الماوس ويمكن أخذها من وسائط أحداث الماوس (e.X و e.Y) لرسم دائرة ، قمنا بتعيين عرض وارتفاع المستطيل بحيث يكونان متساويين - في هذا المثال ، كلاهما 4 بكسل.
- استدعاء التابع Dispose() من الصف graphics يضمن أن يتم إرجاع الموارد الخاصة إلى النظام لإعادة الاستخدام .
- تعتبر الصفوف Graphics و SolidBrush و Color جزءًا من فضاء الحالة System.Drawing .

أحداث لوحة المفاتيح
Keyboard Event Handling

- تحدث الأحداث الرئيسية للمفتاح Keyعند الضغط على مفاتيح لوحة المفاتيح وتحريرها.
- يمكن معالجة مثل هذه الأحداث لأي عنصر تحكم يرث من System.Windows.Forms.Control. ن وهناك ثلاثة أحداث رئيسية :
- KeyPress
- KeyUp
- KeyDown
- يحدث الحدث KeyPress عندما يقوم المستخدم بالضغط على مفتاح يمثل حرف ASCII ويمكن تحديد المفتاح المحدد باستخدام الخاصية KeyChar (e.KeyChar) عن طريق غرض من الصف KeyPressEventArgs الخاصة بمعالج الحدث.

- لا يشير الحدث KeyPress إلى ما إذا كانت مفاتيح المُعدِّل (على سبيل المثال ، Shift و Alt و Ctrl) قد تم الضغط عليها عند حدوث حدث رئيسي.
- إذا كانت هذه المعلومات مهمة ، فيمكن استخدام أحداث KeyUp أو Key-Down ، ويحتوي الوسيط من KeyEventArgs لكل من هذه الأحداث على معلومات حول مفاتيح التعديل modifier keys.
- يوضح الجدول(17) معلومات الأحداث الرئيسية المهمة.
- يوضح البرنامج التالي استخدام معالجات الأحداث الرئيسية لإظهارمفتاح تم الضغط عليه من قبل المستخدم.
- البرنامج عبارة عن Form به لافتتان ، أحدهما تظهر المفتاح المضغوط والأخرة تظهر معلومات عن مفتاح التصنيف والمعدّل .

Keyboard events and event arguments	
Key Events with Event Arguments of Type `KeyEventArgs`	
`KeyDown`	Generated when a key is initially pressed.
`KeyUp`	Generated when a key is released.
Key Event with Event Argument of Type `KeyPressEventArgs`	
`KeyPress`	Generated when a key is pressed. Raised after `KeyDown` and before `KeyUp`.
Class `KeyPressEventArgs` *Properties*	
`KeyChar`	Returns the ASCII character for the key pressed.
Class `KeyEventArgs` *Properties*	
`Alt`	Indicates whether the *Alt* key was pressed.
`Control`	Indicates whether the *Ctrl* key was pressed.
`Shift`	Indicates whether the *Shift* key was pressed.
`KeyCode`	Returns the key code for the key as a value from the `Keys` enumeration. This does not include modifier-key information. It's used to test for a specific key.
`KeyData`	Returns the key code for a key combined with modifier information as a `Keys` value. This property contains all information about the pressed key.
`KeyValue`	Returns the key code as an `int`, rather than as a value from the `Keys` enumeration. This property is used to obtain a numeric representation of the pressed key. The `int` value is known as a Windows virtual key code.
`Modifiers`	Returns a `Keys` value indicating any pressed modifier keys (*Alt*, *Ctrl* and *Shift*). This property is used to determine modifier-key information only.

الجدول(17) | Keyboard events and event arguments.

البرنامج الكامل : أحداث لوحة المفاتيح
Keyboard event

البرنامج

```
using System;

using System.Windows.Forms;

namespace WindowsFormsApplication15
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

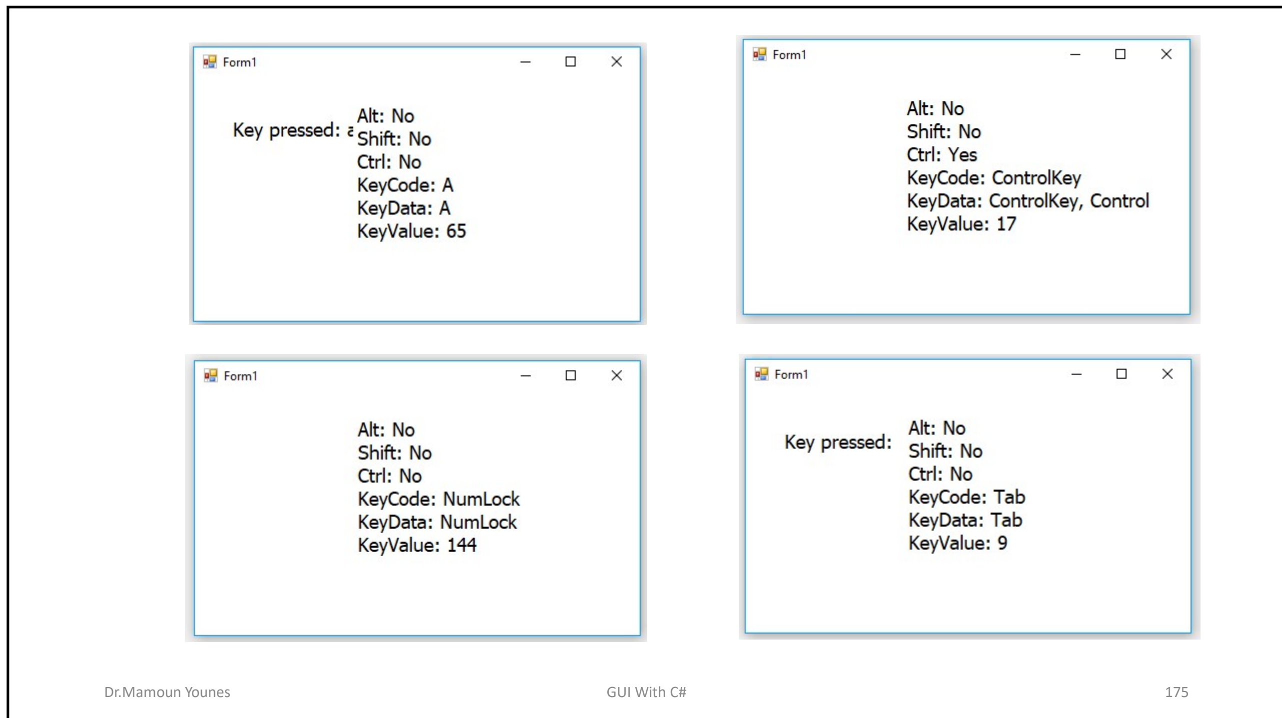
        }
```



```
private void Form1_KeyPress(object sender, KeyPressEventArgs e)
        {
            charLabel.Text = "Key pressed: " + e.KeyChar;
        }

        private void Form1_KeyDown(object sender, KeyEventArgs e)
        {
            keyInfoLabel.Text =
             "Alt: " + ( e.Alt ? "Yes" : "No" ) + '\n' +
             "Shift: " + ( e.Shift ? "Yes" : "No" ) + '\n' +
              "Ctrl: " + ( e.Control ? "Yes" : "No" ) + '\n' +
             "KeyCode: " + e.KeyCode + '\n' +
             "KeyData: " + e.KeyData + '\n' +
             "KeyValue: " + e.KeyValue;
        }

        private void Form1_KeyUp(object sender, KeyEventArgs e)
        {
            charLabel.Text = "";
            keyInfoLabel.Text = "";
        }
    }//end class
}// end namespace
```



- يعرض عنصر التحكم في اللافتة charLabel قيمة الحرف للمفتاح المضغوط ، بينما يعرض keyInfo-Label المعلومات المتعلقة بالمفتاح المضغوط.
- لأن الحدثين KeyDown و KeyPress تنقلان معلومات مختلفة ، ويعالج الـ Form (KeyDemo) كلاهما.
- يصل معالج الأحداث KeyPress (الأسطر 14–17) إلى خاصية KeyChar الخاصة بالغرض KeyPressEventArgs ويؤدي هذا إلى إرجاع المفتاح المضغوط كمحرف char ، ثم نعرضه في اللافتة charLabel (السطر 16).
- إذا لم يكن المفتاح المضغوط عبارة عن حرف ASCII، فلن يحدث حدث KeyPress، ولن تعرض اللافتة charLabel أي نص.

- يعرض معالج الحدث KeyDown (الأسطر 18–27) المعلومات عن غرض من KeyEventArgs الخاص به ، ويقوم معالج الحدث باختبار مفاتيح Alt و Shift و Ctrl باستخدام خصائص Alt و Shift و Control، والتي ترجع كل منها قيمة بولانية true إذا تم الضغط على المفتاح المطابق وإلا سوف يرجع false ، ثم يعرض معالج الحدث خصائص KeyCode و KeyData و KeyValue .
  - ترجع الخاصية KeyCode قائمة بقيمة المفاتيح Keys (السطر 24) ، أما الخاصية KeyCode تقوم بإرجاع المفتاح المضغوط ، ولكنها لا توفر أي معلومات حول مفاتيح التعديل modifier keys ، ويتم تمثيل كل من الحرف الصغير والكبير مثل (a) كمفتاح A .
  - ترجع الخاصية KeyData (السطر 25) أيضاً قائمة بقيمة المفاتيح ولكن هذه الخاصية تتضمن بيانات حول مفاتيح التعديل ، وبالتالي ، إذا كان " A " هو الدخل ، فستعرض KeyData أنه تم الضغط على كل من المفتاح A ومفتاح Shift ، وأخيراً ، يُرجع KeyValue (السطر 26) قيمة int تمثل مفتاحًا مضغوطًا، وهذا int هو رمز المفتاح ، ورمز المفتاح مفيد عند اختبار مفاتيح غير ASCII مثل F12 .

- يمسح معالج الحدث KeyUp (الأسطر 28–32) كل اللافتات عند تحرير المفتاح ،وكما يمكننا أن نرى من الخرج ، لا يتم عرض مفاتيح غير ASCII في اللافتة charLabel ، لأنه لم يتم إنشاء حدث KeyPress، على سبيل المثال ، لا تعرض اللافتة charLabel أي نص عند الضغط على مفاتيح F7 أو Tab ، ومع ذلك ، لا يزال يتم توليد الحدث Key-Down، وتعرض اللافتة keyInfoLabel معلومات حول المفتاح الذي تم الضغط عليه ، ويمكن استخدام لائحة المفاتيح لاختبار مفاتيح محددة عن طريق مقارنة رمز المفتاح للمفتاح المضغوط بالقيم الموجودة في لائحة المفاتيح.

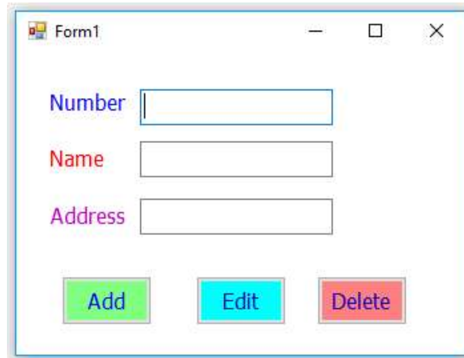
أدوات التلميح
ToolTips

- إن تلميحات الأدوات المعروضة في Visual Studio تساعدنا على التعرف على ميزات IDE ، وتكون بمثابة تذكير مفيد لوظيفة أي رمز من شريط الأدوات، وتستخدم العديد من البرامج نصائح الأدوات لتذكير المستخدمين بكل هدف من عناصر التحكم.
- على سبيل المثال ، يحتوي Microsoft Word على نصائح الأدوات التي تساعد المستخدمين على تحديد الغرض من رموز التطبيق.
- يوضح هذا القسم كيفية استخدام التلميحات لعناصر ToolTip لإضافة معلومات للأدوات في التطبيق .
- يوضح الجدول(18)الخصائص العامة والحدث العام في الصف ToolTip.

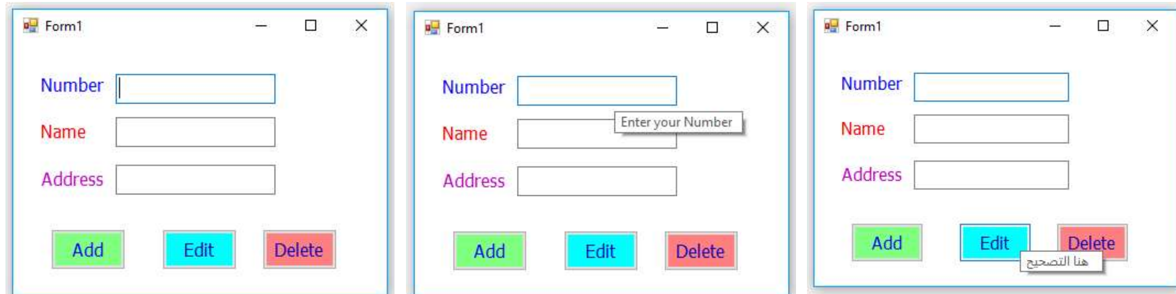
ToolTip properties and an event	Description
Common Properties	
AutoPopDelay	The amount of time (in milliseconds) that the tool tip appears while the mouse is over a control.
InitialDelay	The amount of time (in milliseconds) that a mouse must hover over a control before a tool tip appears.
ReshowDelay	The amount of time (in milliseconds) between which two different tool tips appear (when the mouse is moved from one control to another).
Common Event	
Draw	Raised when the tool tip is displayed. This event allows programmers to modify the appearance of the tool tip.

الجدول(18) | ToolTip properties and an event.

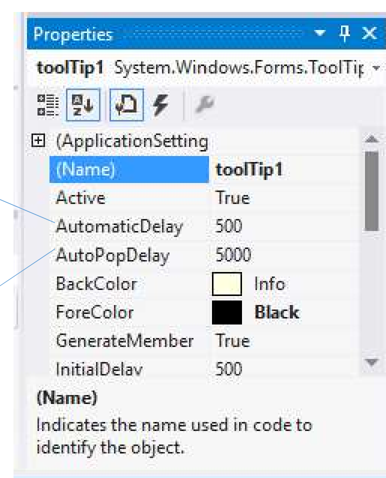
- في المثال التالي :
  - ➢ ننشئ ثلاث لافتات Lable ونكتب بها : Number , Name , Address .
  - ➢ ننشئ ثلاثة صناديق نص TextBox .
  - ➢ ننشئ ToolTip ، فيظهر في أسفل الشاشة .
  - ➢ ننشئ ثلاثة أزرار ونكتب عليها : Add , Edit , delete
- عند تنفيذ البرنامج تظهر الواجهة التالية :



- لنكتب تلميح لكل عنصر كما يلي :
  - ➢ نقف على العنصر TextNumber هو TextBox ونكتب تلميح في الخاصية ToolTip on tool مثلاً العبارة Enter Your Number .
  - ➢ نقف على العنصر TextName هو TextBox ونكتب تلميح في الخاصية ToolTip on tool مثلاً العبارة Enter Your Name .
  - ➢ نقف على العنصر TextAddress هو TextBox ونكتب تلميح في الخاصية ToolTip on tool مثلاً العبارة Enter Your Address .
  - ➢ نقف على العنصر btnAdd هو Button ونكتب تلميح في الخاصية ToolTip on tool مثلاً العبارة (هنا الإضافة) .
  - ➢ نقف على العنصر btnEdit هو Button ونكتب تلميح في الخاصية ToolTip on tool مثلاً العبارة (هنا التصحيح) .
  - ➢ نقف على العنصر btnDelete هو Button ونكتب تلميح في الخاصية ToolTip on tool مثلاً العبارة (هنا الحذف) .

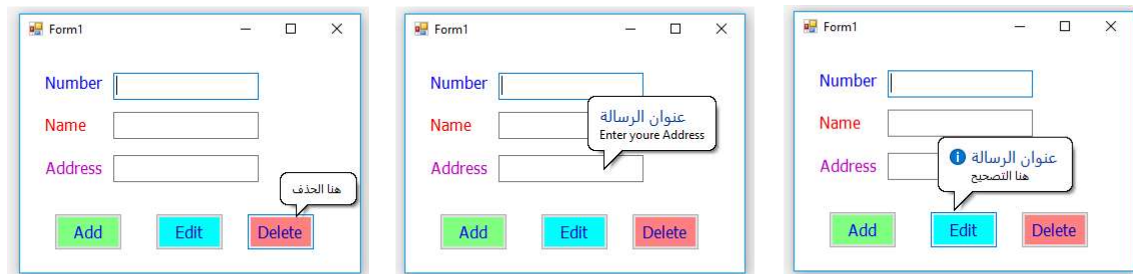


- يدل على الفترة اللازمة لظهور التلميح وهو 500mS
- يمكن تعديل القيمة لـ 2000 ستكون الفترة طويلة للظهور أو أصغر 10 عندها ستكون الفترة سريعة للظهور .



- يدل على الفترة اللازمة لبقاء التلميح حتى يختفي وهو 5000mS
- يمكن تعديل القيمة لـ 9000 أو أصغر 100 عندها ستكون الفترة للازمة الاختفاء سريعة .

- يمكن تغيير شكل التلميح من مستطيل إلى بالون Balloon ، وذلك بتغيير الخاصة IsBalloon من False إلى True .
- يمكن كتابة عبارة ما (عنوان الرسالة) على التلميح من الخاصية ToolTip Title .
- يمكن إضافة أيقونة Icon على التلميح (Error , Warning , Info) من الخاصية ToolTipIcon .
- حتى يكون ظهور واختفاء التلميح بشكل مريح نغير الخاصية UseFading من False إلى True .



البرنامج الكامل : أدوات التلميح ToolTips

البرنامج

```
sing System;
using System.Windows.Forms;

namespace WindowsFormsApplication16
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void labelNumber_Click(object sender, EventArgs e)
        {

        }
```

```
private void textNumber_TextChanged(object sender, EventArgs e)
  {

  }
private void labelName_Click(object sender, EventArgs e)
  {

  }

  private void textName_TextChanged(object sender, EventArgs e)
  {

  }

  private void labelAddress_Click(object sender, EventArgs e)
  {

  }
```



```
        private void textAddress_TextChanged(object sender, EventArgs e)
         {

         }

        private void btnAdd_Click(object sender, EventArgs e)
         {

         }

         private void btnEdit_Click(object sender, EventArgs e)
         {

         }

         private void btnDelete_Click(object sender, EventArgs e)
         {

         }
    }//end class
}// end namespace
```

التحكم بالأرقام
NumericUpDown Control

- في بعض الأحيان ، نحتاج إلى تقييد خيارات إدخال المستخدم لنطاق محدد من القيم الرقمية ،وهذا هو الغرض من عنصر التحكم NumericUpDown .
- يظهر عنصر التحكم هذا على هيئة TextBox، مع وجود زرين صغيرين على الجانب الأيمن - أحدهما به سهم للأعلى وواحد به سهم للأسفل.
- بشكل افتراضي ، يمكن للمستخدم كتابة القيم الرقمية في عنصر التحكم هذا كما لو كان TextBox أو النقر فوق السهمين للأعلى وللأسفل لزيادة أو إنقاص القيمة في عنصر التحكم.
- يتم تحديد أكبر و أصغر القيم في المجال من الخاصية Maximum و Minimum (كلاهما من النوع العشري).
- تحدد خاصية الزيادة Increment(أيضًا من النوع العشري) إلى أي مدى تتغير القيمة الحالية عندما يقوم المستخدم بالنقر فوق الأسهم.
- تحدد الخاصية DecimalPlaces عدد الخانات العشرية التي يجب أن يعرضها عنصر التحكم كعدد صحيح. يوضح الجدول(19)الخصائص العامة والأحداث لـ NumericUpDown .

NumericUpDown properties and an event	Description
Maximum	Largest value in the control's range.
Minimum	Smallest value in the control's range.
UpDownAlign	Modifies the alignment of the up and down Buttons on the NumericUpDown control. This property can be used to display these Buttons either to the left or to the right of the control.
Value	The numeric value currently displayed in the control.
Common Event	
ValueChanged	This event is raised when the value in the control is changed. This is the default event for the NumericUpDown control.

الجدول(19) | NumericUpDown properties and an event. (Part 2 of 2.)

- **NumericUpDown :** لا نستطيع نكتب بها إلا الأرقام ، والشكل الافتراضي لها الأرقام الصحيحة ، فيها الخاصية TextAlign وتأخذ القيمة Left بالشكل الافتراضي ن ويمكن تغيرها إلى Right أو Center ، من خلال السهمين يمكن الزيادة أو النقصان بمقدار 1 وتكون محددة من 0 حتى الـ 100 بشكل افتراضي .
  - من خلال الخاصية DecimalPlaces فهي تحدد عدد الخانات بعد الفاصلة العشرية ونضبطها على الرقم 4 أي أربع خانات بعد الفاصلة العشرية فيظهر الرقم مثلاً 4.4500 أو 0.0000 .
  - من خلال الخاصية HexaDecimal تكون في الحالة False ، ويمكن تغيرها إلى الحالة True ، سوف تعمل بنظام السداسي عشر .
  - من خلال الخاصية Increment نستطيع الزيادة أو النقصان بمقدار 2 أو 3 أو أي عدد آخر ، بحيث نكتب بجانبها مقدار الزيادة كعدد صحيح .
  - من خلال الخاصية Maximum و الخاصية Minimum نستطيع تحديد أكبر وأصغر قيمة لـ NumericUpDown ، بحيث لا يمكن تجاوزها .
  - تحدد الخاصية Value القيمة الابتدائية التي يبدأ منها العنصر NumericUpDown .
  - default

البرنامج الكامل : التحكم بالأرقام NomericUpDown

البرنامج

```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication18
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            numericUpDown1.Value = 50;
        }
```

```
private void numericUpDown1_ValueChanged(object sender, EventArgs e)
{

}
private void button2_Click(object sender, EventArgs e)
{
    MessageBox.Show(numericUpDown2.Value.ToString());
}

private void numericUpDown2_ValueChanged(object sender, EventArgs e)
{

}
private void numericUpDown3_ValueChanged(object sender, EventArgs e)
{

}
private void numericUpDown4_ValueChanged(object sender, EventArgs e)
{

}
```



```
        private void numericUpDown5_ValueChanged(object sender, EventArgs e)
        {

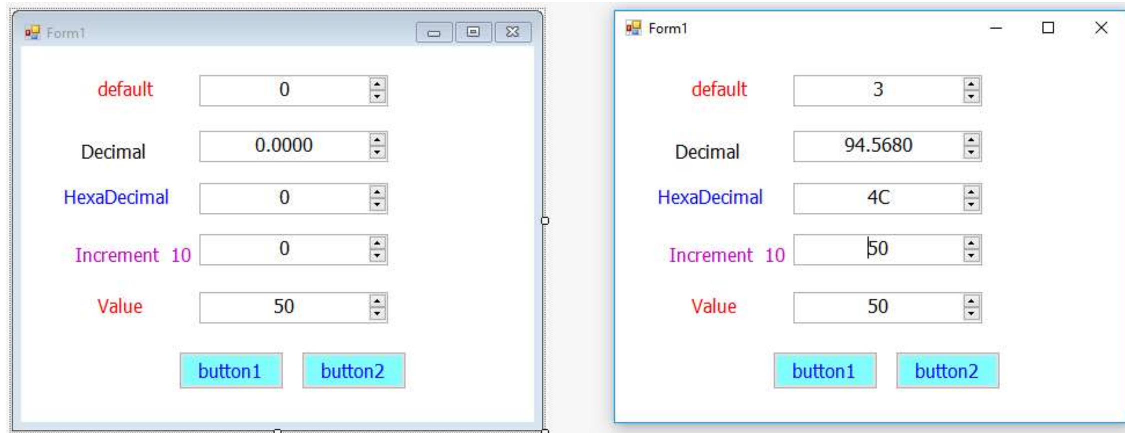
        }
        private void label2_Click(object sender, EventArgs e)
        {

        }
        private void label3_Click(object sender, EventArgs e)
        {

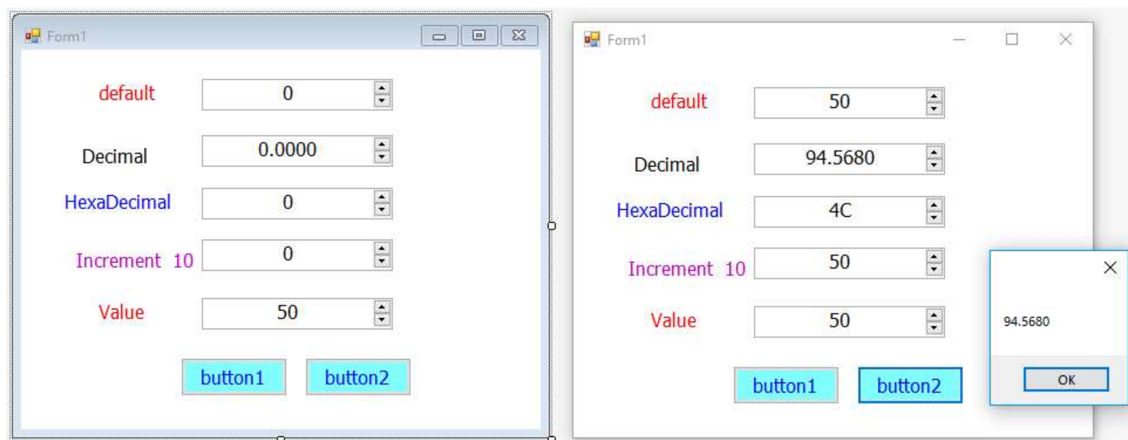
        }
        private void label4_Click(object sender, EventArgs e)
        {

        }
        private void label5_Click(object sender, EventArgs e)
        {

        }
    }//end class
}// end namespace
```



- عند الضغط على الزر الأول سوف يتغير الـ Default من 0 إلى 50 .
- عند الضغط على الزر الثاني سوف يظهر القيمة الموجودة في الـ Decimal.



مثال (1)

- البرنامج التالي يحسب الفائدة المركبة ، لمستثمر يريد أن يودع مبلغ $1000.00في حساب توفير يحقق فائدة 5% ، إذا افتراضنا أن جميع الفوائد متبقية عند الإيداع ، والمطلوب حساب الفائدة المركبة في نهاية كل عام لمدة عشر سنوات ، يتم حساب الفائدة المركبة من العلاقة التالية :

$$a = p\,(1 + r)^n$$

- حيث :
- P: هو المبلغ الأصلي المستثمر (أي المبلغ الرئيسي) (principal) .
- r: هو معدل الفائدة السنوي (rate / 100) .
- n: هو عدد السنوات (yearCounter) .
- a: هو المبلغ المودع في نهاية السنة التاسعة(Amount on Deposit) .
- ويطبع الفائدة والمبلغ في كل سنة على واجهة بيانية (نافذة) ، باستخدام MessageBox.Show .
- يتم استخدام TextBoxes لإدخال المبالغ الرئيسية ومعدلات الفائدة ، ويتم استخدام عنصر تحكم NumericUpDown لإدخال عدد السنوات التي نريد حساب الفائدة.
- يتم استخدام TextBoxes ثالث لطباعة الفائدة والمبلغ في كل سنة .

يوضح هذا المثال عنصر التحكم NumericUpDown فـي GUI الذي يحسب معدل الفائدة interest rate .

البرنامج

```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication17
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void calculateButton_Click(object sender, EventArgs e)
        {
            // declare variables to store user input
          decimal principal; // store principal
          double rate; // store interest rate
         int year; // store number of years
         decimal amount; // store amount
        string output; // store output
```

```
// retrieve user input
  principal = Convert.ToDecimal( principalTextBox1.Text );

      rate = Convert.ToDouble( interestTextBox2.Text );
   year = Convert.ToInt32( yearUpDown.Value );

  // set output header
 output = "Year\tAmount on Deposit\r\n";

   // calculate amount after each year and append to output
  for ( int yearCounter = 1; yearCounter <= year; ++yearCounter )
   {
       amount = principal *
      ( ( decimal ) Math.Pow( ( 1 + rate / 100 ), yearCounter ) );
      output += ( yearCounter + "\t " +
      string.Format( "{0:C}", amount ) + "\r\n" );
     } // end for

  displayTextBox3.Text = output; // display result

  }//  end calculateButton
```



```
private void principalTextBox1_TextChanged(object sender, EventArgs e)
 {

 }

 private void label1_Click(object sender, EventArgs e)
 {

 }

 private void interestTextBox2_TextChanged(object sender, EventArgs e)
 {

 }

 private void label2_Click(object sender, EventArgs e)
 {

 }
```

```
        private void yearUpDown_ValueChanged(object sender, EventArgs e)
        {

        }

        private void label3_Click(object sender, EventArgs e)
        {

        }

        private void label4_Click(object sender, EventArgs e)
        {

        }

        private void displayTextBox3_TextChanged(object sender, EventArgs e)
        {

        }
    }//end class
}// end namespace
```


Form1

Principal

Calculate

Interest Rate

Year 0

Yearly Account Balance

Form1

Principal 1000

Calculate

Interest Rate 5

Year 10

Yearly Account Balance

Year	Amount on Deposit
1	£1,050.00
2	£1,102.50
3	£1,157.63
4	£1,215.51
5	£1,276.28
6	£1,340.10

عنصر التحكم TabControl

- يُعد عنصر التحكم TabControl أحد أنواع الحاويات ، تتألف الحاوية من عدة بوابات نستطيع ان نصنع في كل عنصر مجموعة من العناصر .
- تحتوي TabControls على أغراض TabPage والتي تشبه الحاويات Panels و GroupBoxes وتحتوي على عناصر تحكم .
- يمكننا إضافة عناصر تحكم إلى أغراض TabPage ، ثم إضافة TabPages إلى TabControl ، ويتم عرض TabPage واحد في كل مرة .
- **يتم إضافة أغراض TabPage و TabControls كما يلي :**

- *myTabPage*.Controls.Add(*myControl*);
- *myTabControl*.TabPages.Add(*myTabPage*);

- يستدعي التابع التعليمات السابقة ويضيفها إلى تجمع عناصر التحكم وإلى تجمع TabPage
- **في المثال يضيف TabControl هو *myControl* إلى TabPage هو *myTabPage* ثم يضيف *myTabControl* إلى *myTabPage* .**
- **بدلاً من ذلك ، يمكننا استخدام التابع AddRange** لإضافة مصفوفة من TabPages أو عناصر التحكم إلى TabControl أو TabPage على التوالي .
- **يمكننا إضافة TabControls بشكل مرئي عن طريق السحب والإفلات إلى النموذج Form في وضع Design، ولإضافة TabPages في وضع Design، انقر فوق الجزء العلوي من TabControl، وافتح قائمة المهام الذكية الخاصة به *smart tasks menu* واختر Add Tab .**
- بدلاً من ذلك ، انقر فوق خاصية TabPages في قائمة الخصائص Properties وأضف tabs في مربع الحوار الذي يظهر. ولتغيير تسمية tabs ، قم بتعيين الخاصية Text لـ TabPage ، أما النقر فوق الـ tabs لاختيار TabControl ثم اختيار TabPage بالنقر فوق منطقة التحكم أسفل tabs .
- يمكن إضافة عناصر تحكم إلى TabPage عن طريق سحب العناصر وإفلاتها من ToolBox لعرض TabPages مختلفة .
- يوضح الجدول (20) خصائص TabControl وتابع الحدث .

TabControl properties and an event	Description
Common Properties	
ImageList	Specifies images to be displayed on tabs.
ItemSize	Specifies the tab size.
Multiline	Indicates whether multiple rows of tabs can be displayed.
SelectedIndex	Index of the selected TabPage.
SelectedTab	The selected TabPage.
TabCount	Returns the number of tab pages.
TabPages	Returns the collection of TabPages within the TabControl as a TabControl.TabPageCollection.
Common Event	
SelectedIndexChanged	Generated when SelectedIndex changes (i.e., another TabPage is selected).

الجدول(20) | TabControl properties and an event.

مثال على عنصر التحكم TabControll

- البرنامج التالي يقوم بإنشاء :

1. أربعة بوابات Tabs ، الأولى Color والثانية Size والثالثة Message والرابعة About .
2. ثلاثة أزرار RadioButtons للبوابة الأولى ، الأول Black والثاني Red والثالث Green .
3. ثلاثة أزرار RadioButtons للبوابة الثانية ، الأول 12Point والثاني 16Point والثالث 30Point .
4. زران RadioButtons للبوابة الثالثة ، الأول Hello والثاني Goodbye .
5. مربع نص textBox للبوابة الرابعة من أجل لإظهار نص حول عنصر التحكم Tabs .
6. إنشاء لافتة Label من أجل إظهار رسالة ترحيب Hello أو وداع Goodbye وتلوينها من البوابة الأولى وتغير حجم الخط من البوابة الثانية .

البرنامج

```
using System;

using System.Drawing;

using System.Windows.Forms;

namespace Tabs2
{
    public partial class UsingTabsForm : Form
    {
        public UsingTabsForm()
        {
            InitializeComponent();
        }

        private void blackRadioButton_CheckedChanged(object sender, EventArgs e)
        {
            displayLabel.ForeColor = Color.Black; // change color to black
        }
```



```
private void redRadioButton_CheckedChanged(object sender, EventArgs e)
 {
   displayLabel.ForeColor = Color.Red; // change color to red
 }
private void greenRadioButton_CheckedChanged(object sender, EventArgs e)
 {
   displayLabel.ForeColor = Color.Green; // change color to green
 }

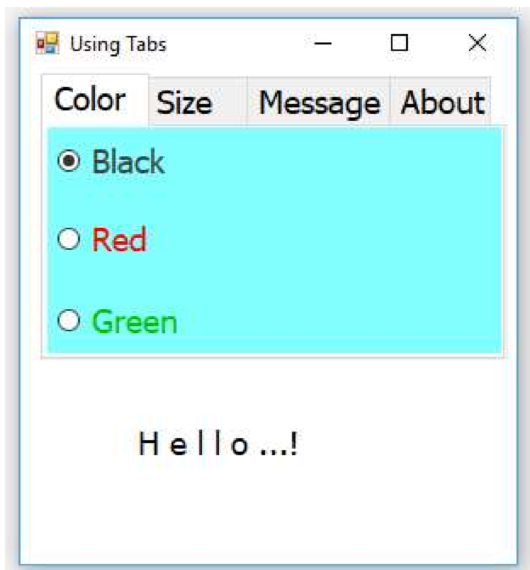
 private void size12RadioButton_CheckedChanged(object sender, EventArgs e)
 {
     // change font size to 12
   displayLabel.Font = new Font(displayLabel.Font.Name, 12);
 }

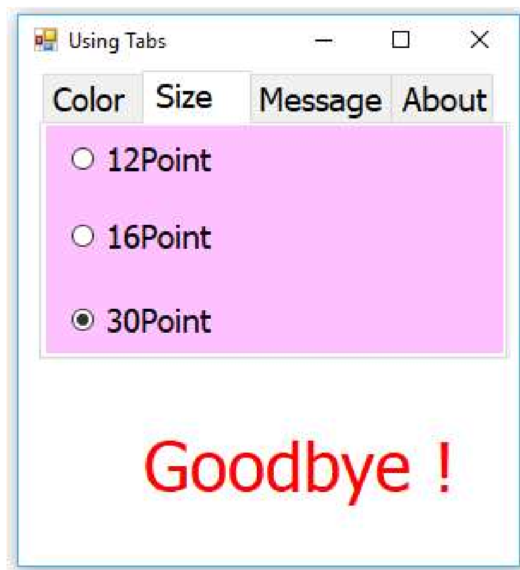
private void size16RadioButton_CheckedChanged(object sender, EventArgs e)
 {
   // change font size to 16
   displayLabel.Font = new Font(displayLabel.Font.Name, 16);
 }
```

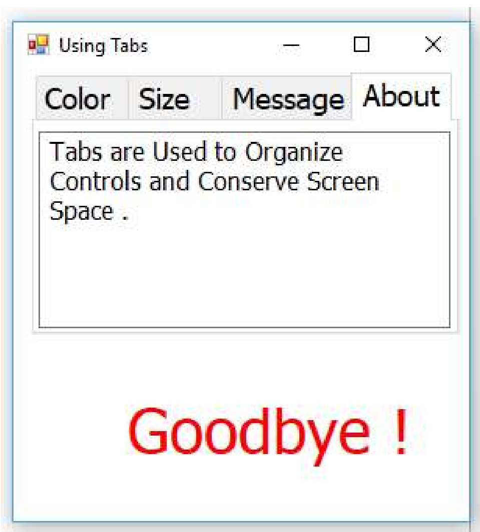
```
private void size20RadioButton_CheckedChanged(object sender, EventArgs e)
  {
      // change font size to 20
      displayLabel.Font = new Font(displayLabel.Font.Name, 30);
  }

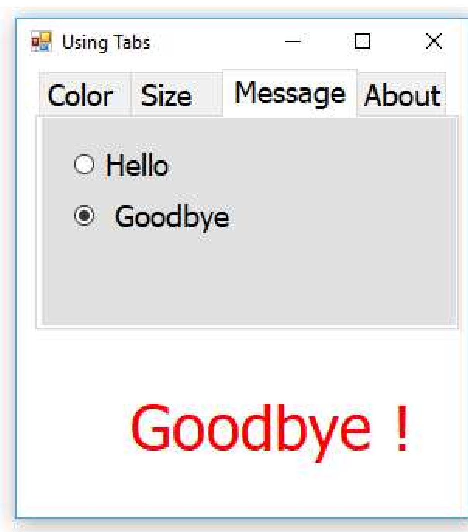
 private void helloRadioButton_CheckedChanged(object sender, EventArgs e)
  {
      displayLabel.Text = "Hello !"; // change text to Hello!
  }

 private void goodbyeRadioButton_CheckedChanged(object sender, EventArgs e)
  {
      displayLabel.Text = "Goodbye !"; // change text to Goodbye!
  }
 }// end class UsingTabsForm
}// end namespace UsingTabs
```







مثال ثاني على عنصر التحكم TabControll

- **المثال التالي هو تعديل على المثال السابق** ، بالإضافة إلى أربعة بوابات التي أنشأناها في المثال السابق ، فسوف **نضيف بوابة خامسة اسمها Figures** ، تحتوي على أربعة أزرار كما يلي :
  1. **الزر الأول هو Rectangle** لرسم مستطيل مصمت لونه أحمر .
  2. **الزر الثاني هو FillRectangle** لرسم مستطيل ممتلئ لونه أزرق .
  3. الزر الثالث هو Oval لرسم قطع ناقص مصمت لونه زهري .
  4. **الزر الرابع هو FillOval** لرسم قطع ناقص ممتلئ لونه Cyan.

- البرنامج التالي يحتوي على خمس بوابة لكل بوابة وظيفة محددة كما يلي :

البرنامج

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace TabsFigure
{
    public partial class UsingTabsForm : Form
    {
        public UsingTabsForm()
        {
            InitializeComponent();
        }

        private void blackRadioButton_CheckedChanged(object sender, EventArgs e)
        {
            displayLabel.ForeColor = Color.Black; // change color to black
        }

        private void redRadioButton_CheckedChanged(object sender, EventArgs e)
        {
            displayLabel.ForeColor = Color.Red; // change color to red
        }
```



```
       private void greenRadioButton_CheckedChanged(object sender, EventArgs e)
        {
            displayLabel.ForeColor = Color.Green; // change color to green
        }

        private void size12RadioButton_CheckedChanged(object sender, EventArgs e)
        {
            // change font size to 12
            displayLabel.Font = new Font(displayLabel.Font.Name, 12);
        }

        private void size16RadioButton_CheckedChanged(object sender, EventArgs e)
        {
            // change font size to 16
            displayLabel.Font = new Font(displayLabel.Font.Name, 16);
        }

        private void size20RadioButton_CheckedChanged(object sender, EventArgs e)
        {
            // change font size to 20
            displayLabel.Font = new Font(displayLabel.Font.Name, 30);
        }
```

```
private void helloRadioButton_CheckedChanged(object sender, EventArgs e)
 {
     displayLabel.Text = "Hello !"; // change text to Hello!
 }

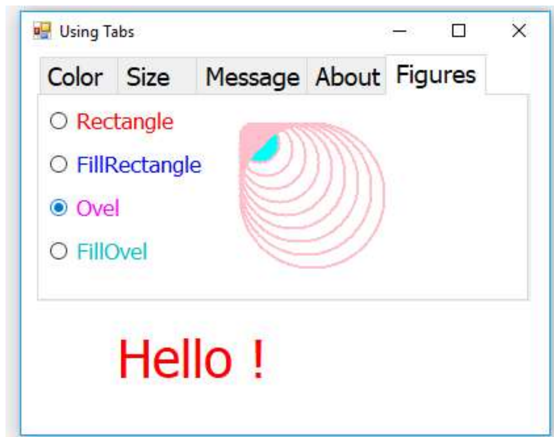
 private void goodbyeRadioButton_CheckedChanged(object sender, EventArgs e)
 {
     displayLabel.Text = "Goodbye !"; // change text to Goodbye!
 }

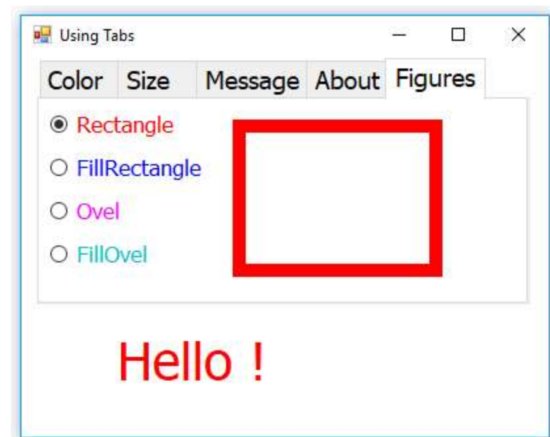
private void radioButton1_CheckedChanged(object sender, EventArgs e)
 {
     tabPage5.CreateGraphics().Clear(Color.White);
     Pen pp = new Pen(Color.Red, 10);
     tabPage5.CreateGraphics().DrawRectangle(pp, 150, 20, 150, 110);
 }
private void radioButton2_CheckedChanged(object sender, EventArgs e)
 {
     tabPage5.CreateGraphics().Clear(Color.White);
     tabPage5.CreateGraphics().FillRectangle(Brushes.Blue, 150, 20, 150, 110);
 }
```



```
private void radioButton3_CheckedChanged(object sender, EventArgs e)
   {
      tabPage5.CreateGraphics().Clear(Color.White);
      tabPage5.CreateGraphics().FillEllipse(Brushes.Cyan, 150, 20, 30, 30);
      Pen pp = new Pen(Color.Pink, 2);
      for(int i=0;i<10;i++)
      tabPage5.CreateGraphics().DrawEllipse(pp, 150, 20, 20 + 10 * i,20 + 10 * i);
   }

private void radioButton4_CheckedChanged(object sender, EventArgs e)
   {
      tabPage5.CreateGraphics().Clear(Color.White);
      tabPage5.CreateGraphics().FillEllipse(Brushes.Cyan, 150, 20, 150, 110);
     }
  }// end class UsingTabsForm
}// end namespace UsingTabs
```





Using Tabs

Color Size Message About Figures

Rectangle

FillRectangle

Ovel

FillOvel

Hello !

Using Tabs

Color Size Message About Figures

Rectangle

FillRectangle

Ovel

FillOvel

Hello !

التحكم بالمؤقت Timer
Timer Control

- يقوم العنصر **Timer** بتكرار عمل ما وذلك عند انقضاء فترة زمنية محددة .
- هذه الأداة لا توضع ضمن النموذج Form وإنما توضع أسفل النموذج .
- أهم خصائصها :
  - ✓ الخاصية Interval : تحدد الفترة الزمنية اللازم مرورها لتكرار تنفيذ عمل المؤقت ، وتقدر mS .
  - ✓ الخاصية Enabled : تحدد فيما إذا المؤقت في حالة تأهيل أم لا ، وله قيمتان :
    - ➢ False : يكون المؤقت غير مؤهلاً .
    - ➢ True : يكون المؤقت غير مؤهلاً .

تابع معالج الحدث هو Tick الذي ينفذ مجموعة من التعليمات الخاصة بالمؤقت عند كل مرور للفترة الزمنية المحددة بالخاصية Interval .

مثال على التحكم المؤقت Timer

- إنشاء غرض من الصف TextBox كما يلي :

```
TextBox txt = new TextBox()
this.Controls.Add(txt);
```

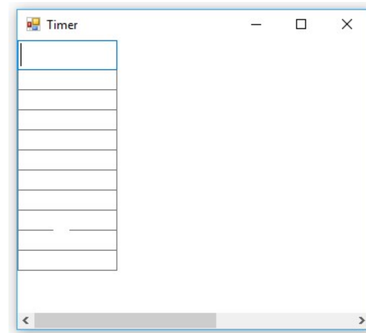
يتم لإنشاء مربع نص واحد TextBox .

- إذا أردنا انشاء عدد محدد من مربعات النص يكون تابع الحدث كما يلي :

```
private void myTtimer1_Tick(object sender, EventArgs e)
   {

        TextBox txt = new TextBox();
        txt.Top = myTop;
        if (txt.Top >= 200)
         myTtimer1.Enabled = false;
         this.Controls.Add(txt);
          myTop += 20;
}
```

عند تنفيذ البرنامج يظهر الخرج التالي

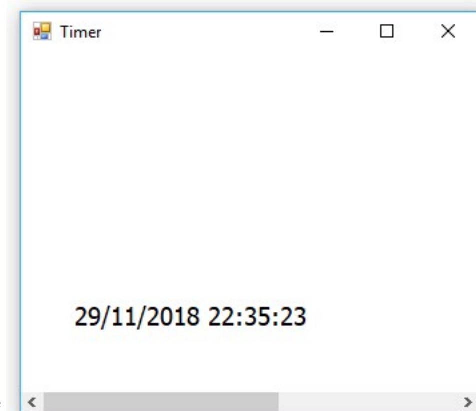


يـتم التوقـف عنـدما يصل Pixel 200
ويتكرر مربع الحوار كل 20 ميللي ثانيـة
.

- انشاء لافتة Label من أجل وضع التاريخ والوقت الخالي داخل اللافتة كما يلي :

```
private void myTtimer1_Tick(object sender, EventArgs e)
   {
        TextBox txt = new TextBox();
         lbl.Text = DateTime.Now.ToString();
}
```

عند تنفيذ البرنامج يظهر الخرج التالي

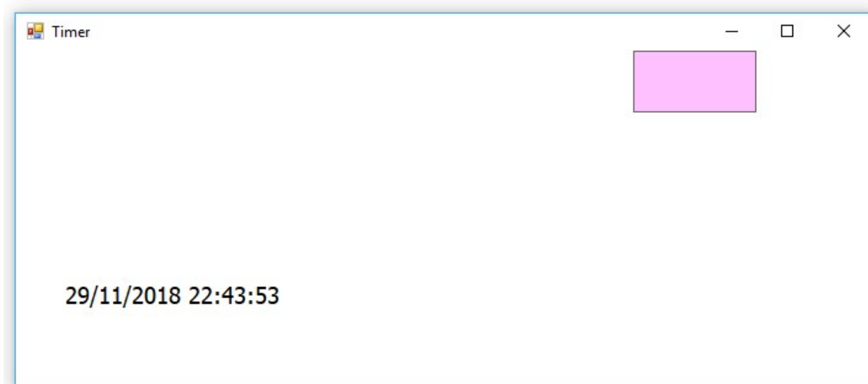


يتم إظهار الوقت الحالي ويبدأ بالعد كل ثانية كأنه ساعة .

- يتم إنشاء صندوق الصورة PictureBox ونضعه في الزاوية اليسارية العليا أي : x=0 , y=0 ، من الخاصية Location ، ثم تحريك الصورة بشكل أفقي بفترة زمنيي صغيرة وحتى لا تخرج الصورة خارج النموذج يتم تحديده كما هو مبين في تابع الحدث :

```
private void myTtimer1_Tick(object sender, EventArgs e)
   {
         TextBox txt = new TextBox();
          lbl.Text = DateTime.Now.ToString();
         if (pictureBox.Left >= 500)
             myTtimer1.Enabled = false;
              pictureBox1.Left += 1;
   }
```


عند تنفيذ البرنامج يظهر الخرج التالي



يتم التوقف عندما يصل 500 Pixel ويتوقف مؤقت الزمن عندما تتوقف حركة الصورة .

مثال ثاني على التحكم المؤقت Timer

- يقوم المثال التالي بالتعرف على كيفية استخدام العنصر Timer ، نلاحظ أن لون العبارة One for All & All for One سيتغير كل ثانية ، ويعود هذا إلى استخدام غرض من timer الصف Timer ، وإن أغراض هذا الصف مسؤولة عن تكرار عمل معين بعد انقضاء فترة زمنية معينة .
- يتم لإنشاء لافتة Label من أجل كتابة النص الواجب تغير لون العبارة One for All & All for One كل واحد ثانية بواسطة الصف Random كما يلي :

```
Random  random1 = new Random( );
int r1 = random1.Next( ) % 250 ;
int g1 = random1.Next( ) % 250;
int b1 = random1.Next( ) % 250;
int a1 = random1.Next() % 225;
Color  c1 = Color.FromArgb(a1, r1, g1, b1);
```


- إنشاء لافتة ثانية Label من أجل كتابة النص الواجب تغير لون العبارة " السلام عليكم ورحمة الله وبركاته" كل واحد ثانية بواسطة الصف Random .

```
Random random2 = new Random();
int r2 = random2.Next() % 200;
int g2 = random2.Next() % 200;
int b2 = random2.Next() % 200;
int a2 = random2.Next() % 225;
Color c2 = Color.FromArgb(a2, r2, g2, b2);
```

- **في هذا المثال :**

أولاً : سوف يتم رسم السلسلة على النافذة Form ، بواسطة التابع DrawString ثم نغير لونها بواسطة الـ Timer ، كما يلي :

```
CreateGraphics().DrawString(str2, myFont2, brush2, point2);
```

ثانياً : سوف يتم تغير لون العبارة الموجودة في اللافتة بواسطة الـ Timer كما يلي :

```
lbl2.ForeColor = Color.FromArgb(200 , r1, g1, b1);
```

- **وبالتالي سوف يكون لدينا سلسلتين مرسومتين على النافذة Form ، وسلسلتين في اللافتتين يتم تطبيق عليهم المؤقت Timer كما هو موضح في البرنامج التالي :**

البرنامج

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace TimerJava
{
    public partial class TimerForm : Form
    {
        public TimerForm()
        {
            InitializeComponent();
        }

        private void myTimer_Tick(object sender, EventArgs e)
        {
             Random  random1 = new Random( );
             int r1 = random1.Next( ) % 250 ;
             int g1 = random1.Next( ) % 250;
             int b1 = random1.Next( ) % 250;
             int a1 = random1.Next() % 225;
```

```
        Color  c1 = Color.FromArgb(a1, r1, g1, b1);
        SolidBrush brush = new SolidBrush(c1);
        string str = lbl2.Text;
        Font myFont = new Font(" Tahoma ", 20, FontStyle.Bold);
        Point point = new Point(40, 30);
        CreateGraphics().DrawString(str, myFont, brush, point);
   //-----------------------------------------------
        Random random2 = new Random();
        int r2 = random2.Next() % 200;
        int g2 = random2.Next() % 200;
        int b2 = random2.Next() % 200;
        int a2 = random2.Next() % 225;
        Color c2 = Color.FromArgb(a2, r2, g2, b2);
        SolidBrush brush2 = new SolidBrush(c2);
        string str2 = lbl1.Text;
        Font myFont2 = new Font(" Tahoma ", 20, FontStyle.Bold);
        Point point2 = new Point(40, 100);
        CreateGraphics().DrawString(str2, myFont2, brush2, point2);
```



```
      lbl2.ForeColor = Color.FromArgb(200 , r1, g1, b1);
      lbl2.Font = new Font("Times New Roman", 20, lbl2.Font.Style | FontStyle.Italic);
    //---------------------------------------------------
      lbl1.ForeColor = Color.FromArgb(200, r2, g2, b2); ;
      lbl1.Font = new Font("Times New Roman", 20, lbl1.Font.Style );

      } // end Timer
  }// end class
}// end namespace
```





التحكم بالتقويم الشهري
MonthCalendar Control

- يجب على العديد من التطبيقات إجراء حسابات التاريخ والوقت.
- توفر .NET Framework عنصري تحكم يسمحان للتطبيق باسترجاع معلومات التاريخ والوقت - عنصري التحكم هما : MonthCalendar و DateTimePicker .
- يعرض عنصر التحكم MonthCalendar تقويمًا شهريًا في الـ Form.
- يمكن للمستخدم تحديد تاريخ من الشهر المعروض حاليًا أو يمكنه استخدام الأسهم للانتقال إلى شهر آخر.
- عندما يتم تحديد تاريخ ، يتم تمييزه ، ويمكن تحديد تواريخ متعددة من خلال النقر على التواريخ في التقويم أثناء الضغط باستمرار على مفتاح Shift .
- معالج الحدث الافتراضي لعنصر التحكم هذا هو الحدث DateChanged، الذي يتم تكوينه عند تحديد تاريخ جديد.
- يتم توفير الخصائص التي تسمح لنا بتعديل مظهر التقويم ، وعدد التواريخ التي يمكن اختيارها في وقت واحد ، والحد الأدنى للتاريخ والحد الأقصى للتاريخ الذي يمكن تحديده.
- يوضح الجدول(20) خصائص MonthCalendar ومعالج الحدث العام MonthCalendar.

MonthCalendar properties and an event	Description
MonthCalendar Properties	
FirstDayOfWeek	Sets which day of the week is the first displayed for each week in the calendar.
MaxDate	The last date that can be selected.
MaxSelectionCount	The maximum number of dates that can be selected at once.
MinDate	The first date that can be selected.
MonthlyBoldedDates	An array of dates that will displayed in bold in the calendar.
SelectionEnd	The last of the dates selected by the user.
SelectionRange	The dates selected by the user.
SelectionStart	The first of the dates selected by the user.
Common MonthCalendar Event	
DateChanged	Generated when a date is selected in the calendar.

الجدول(20) | MonthCalendar properties and an event.

التحكم بالتاريخ والوقت
DateTimePicker Control

- يشبه عنصر التحكم DateTimePickerعنصر التحكم MonthCalendar ولكنه يعرض التقويم عند تحديد السهم للأسفل.
- يمكن استخدام DateTimePicker لاسترجاع معلومات التاريخ والوقت من المستخدم.
- تخزن خاصية Value من عنصر التحكم ValueTimePicker غرض من الصف DateTime، الذي يحتوي دائمًا على معلومات التاريخ والوقت.
- يمكننا استرجاع معلومات التاريخ من خلال غرض من الصف DateTime باستخدام خاصية Date، ولا يمكننا استرجاع معلومات الوقت إلا باستخدام خاصية TimeOfDay .
- يعتبر DateTimePicker أيضًا أكثر قابلية للتخصيص من عنصر تحكم - MonthCalendar .
- يحدد تنسيق الخاصية خيارات تحديد المستخدم باستخدام لائحة الخصائص لعنصر التحكم DateTimePickerFormat .

- القيم الموجودة في هذه اللائحة هي : Long تعرض التاريخ بتنسيق طويل ، مثلاً (Thursday, July 10, (2013) ، Short يعرض التاريخ بصيغة قصيرة ، مثلاً (7/10/2013) ، Timeيعرض قيمة زمنية ، مثلاً (5:31:02 م) وCustom (يشير إلى أنه سيتم استخدام تنسيق مخصص).
- إذا تم استخدام قيمة مخصصة Custom ، يتم تحديد الـ display في DateTimePicker باستخدام الخاصية CustomFormat .
- الحدث الافتراضي لعنصر التحكم هذا هو ValueChanged، والذي يحدث عند تغيير القيمة المحددة (سواء كان تاريخًا أم وقتًا)، ويوضح الجدول(21) خصائص DateTimePicker والحدث العام .

DateTimePicker properties and an event	Description
DateTimePicker Properties	
CalendarForeColor	Sets the text color for the calendar.
CalendarMonthBackground	Sets the calendar's background color.
CustomFormat	Sets the custom format string for the date and/or time displayed in the control.
Format	Sets the format of the date and/or time used for the date and/or time displayed in the control.
MaxDate	The maximum date and time that can be selected.
MinDate	The minimum date and time that can be selected.
ShowCheckBox	Indicates if a CheckBox should be displayed to the left of the selected date and time.
ShowUpDown	Indicates whether the control displays up and down Buttons. Helpful when the DateTimePicker is used to select a time—the Buttons can be used to increase or decrease hour, minute and second.
Value	The data selected by the user.
Common DateTimePicker Event	
ValueChanged	Generated when the Value property changes, including when the user selects a new date or time.

الجدول(21) | DateTimePicker properties and an event.

مثال على التحكم بالوقت والتاريخ والتقويم
MonthCalendar and DateTimePicker Control

البرنامج

```
using System;

using System.Windows.Forms;

namespace WindowsFormsApplication19
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void dtp_ValueChanged(object sender, EventArgs e)
        {
            lbl2.Text = dtp.Value.ToString();
        }
```



```
        private void btnshowHide_Click(object sender, EventArgs e)
        {
            if (!myMC.Visible)
            {
                myMC.Visible = true;
                btnshowHide.Text = "-";
            }
            else
            {
                myMC.Visible = false;
                btnshowHide.Text = "+";
            }
        }
        private void myMC_DateChanged(object sender, DateRangeEventArgs e)
        {
            // DateTime dt = myMC.SelectionRange.Start;
            //DateTime dt = myMC.SelectionRange.Start;
            DateTime dt1 = myMC.SelectionRange.Start;
            DateTime dt2 = myMC.SelectionRange.End;

            lbl.Text = dt1.ToString() + "\n" + dt2.ToString();
        }
```

```
        private void lbl2_Click(object sender, EventArgs e)
        {

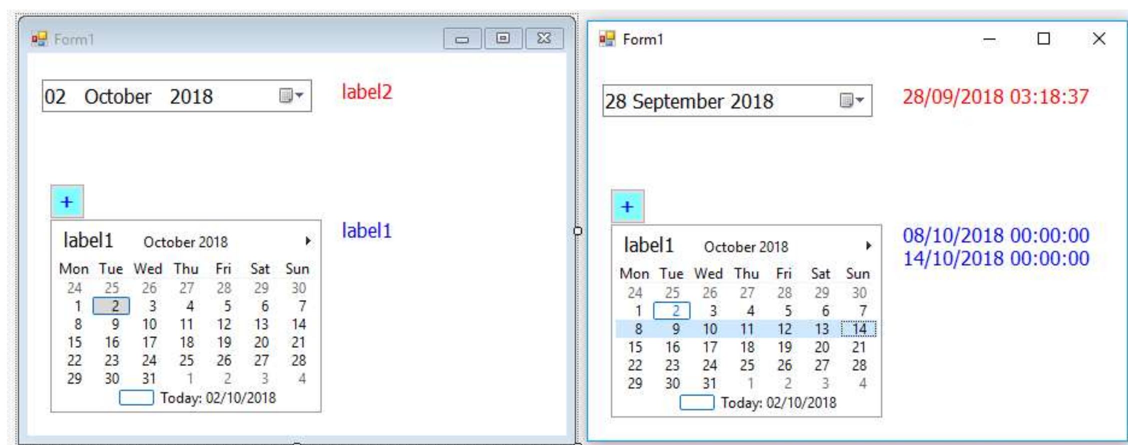
        }

        private void lbl_Click(object sender, EventArgs e)
        {

        }

    }//end class
}// end namespace
```

Start : بداية مجال التاريخ
End : نهاية مجال التاريخ
الخاصية SelectionRange هي التي تحدد المجال وبدونها فإن Start و End لهما نفس الوظيفة .



البرنامج

```
using System;

using System.Windows.Forms;

namespace WindowsFormsApplication20
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
        {
            DateTime dropOffDate = dateTimePicker1.Value ;

       // add extra time when items are dropped off around Sunday
       if ( dropOffDate.DayOfWeek == DayOfWeek.Friday ||
       dropOffDate.DayOfWeek == DayOfWeek.Saturday ||
       dropOffDate.DayOfWeek == DayOfWeek.Sunday )
```

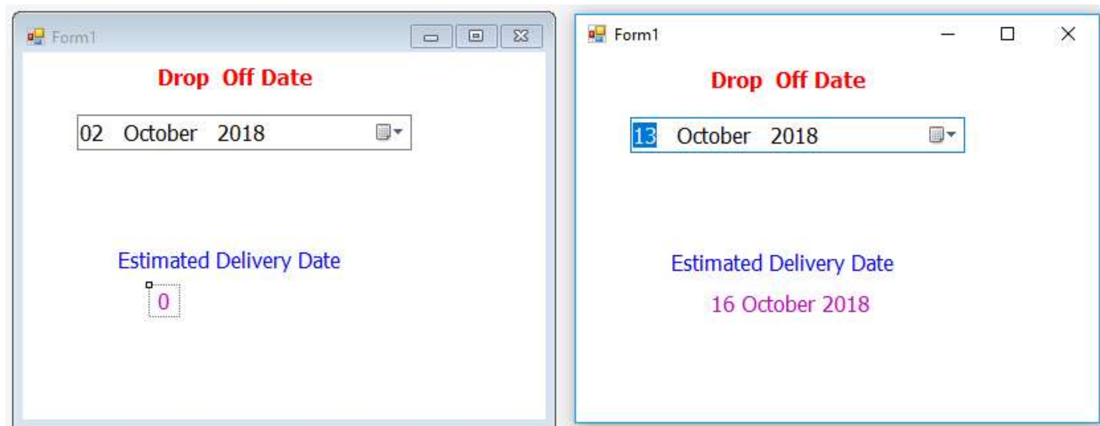


```
      //estimate three days for delivery
        outputLabel.Text =
        dropOffDate.AddDays( 3 ).ToLongDateString();
       else
       // otherwise estimate only two days for delivery
        outputLabel.Text =
       dropOffDate.AddDays( 2 ).ToLongDateString();
     }// end method dateTimePickerDropOff_ValueChanged

         private void Form1_Load(object sender, EventArgs e)
         {
             // user cannot select days before today
             dateTimePicker1.MinDate = DateTime.Today;
             // user can only select days up to one year in the future
             dateTimePicker1.MaxDate = DateTime.Today.AddYears(1);
         }

         private void outputLabel_Click(object sender, EventArgs e)
         {

         }
     }//end class
 }// end namespace
```



the end

The End

The End

The End The End

The End

The End

The End

البرمجة بلغة C#

Graphics Programming

Chapter 3

Dr.Younes

Dr.Mathkoor

Introducing Object-Oriented Design with the UML

الصف Graphics

- تزودنا C# من خلال الصف Graphics بعدد من التوابع التي تسمح لنا برسم صورة معينة أو شكل هندسي ما (خط ، مستطيل ، قطع ناقص ، متعدد الأضلاع , . . . الخ) على مكون ما .
- **يوجد نوعان من التوابع :**
  - الأول يسمح لنا برسم حدود الشكل الهندسي فقط (شكل غير مصمت)
  - الثاني يسمح لنا برسم الشكل الهندسي مصمت(ممتلئ).
- ومن الجدير بالذكر أن مبدأ الإحداثيات هو الزاوية اليسارية العليا لمنطقة الرسم ، مع العلم بأن اتجاه محوري الرسم موضحان في الشكل التالي :



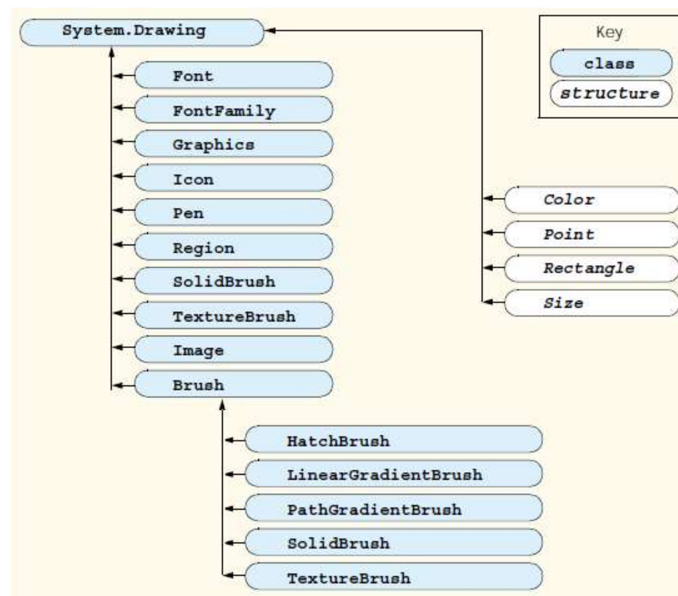
- عند استخدام توابع الرسم يوجد عدد من الوسطاء التي يجب استخدامها ، ومن هذه الوسطاء **نقطة بدء الرسم** مثلاً والتي تمثل الزاوية اليسارية العليا من العنصر وتتزايد قيمة محور السينات x من اليسار إلى اليمين وقيمة محور العينات y من الأعلى إلى الأسفل .
- كما علينا أن نحدد لون خط الرسم قبل استدعاء التابع الذي سيقوم برسم الشكل .
- حيث يتم تحديد **لون خط الرسم** من خلال الخاصية Pens، مثلاً : Pens.Blue ، أما إذا كنا نريد ملء الشكل الهندسي نستخدم الخاصية فرشاة Brushes مثلاً : Brushes.RoyalBlue .
- أيضاً يمكننا أن نحدد **شكل خط الرسم** هل هو مستمر أم متقطع .
- تملك جميع توابع الرسم بارامترات تحدد أماكن رسم الأغراض .

- يتم وضع شيفرة الرسم عادة ضمن الحدث Paint للنافذة Form ، ويتلقى الحدث Paint وسيطاً من النوع PaintEventArgs ، الذي يتضمن غرض من الصف Graphics من أجل النافذة Form التي قامت بقدح الحدث ، و بإمكاننا أخذ غرض من الصف Graphics من أجل إنجاز أية أعمال رسم ضرورية في التطبيق ، وتابع معالج الحدث له الشكل التالي :

```
private void Graphics_Paint(object sender, PaintEventArgs e)
    {

    }
```

- **يوضح الشكل (1)** جزءاً من التسلسل الهرمي للصف System.Drawing، والذي يتضمن عدد من الصفوف الأساسية للرسم البياني وبنيتها .
- **المكونات الأكثر استخداماً في واجهة الرسم البياني** موجودة في الفضاء System.Drawing و System.Drawing.Drawing2D .
- **يحتوي الصف *Graphics*** على التوابع المستخدمة في رسم السلاسل والخطوط والمستطيلات والأشكال الأخرى.
- **تتطلب عادةً توابع الرسم للصف Graphics** غرض من الصف Pen (قلم)أو الصف Brush (فرشاة) لرسم شكل محدد، حيث يُستخدم القلم لرسم الخطوط عريضة الشكل ؛ بينما الفرشاة لرسم الأشياء المصمتة مثل : string و FillRectangle و ......الخ .



الشكل(1) System.Drawing namespace's classes and structures.

الصف Color
Color Class

- يتم إنشاء الألوان من الألوان الثلاثة الرئيسية التالية : BLUE , GREEN , RED ، حيث يتم تمثيل كل منهم باستخدام Byte يصف شدته (والتي تتراوح من القيمة 0 خيال غامق إلى القيمة 255 خيال كاشف) ، يعرف هذا النموذج اللوني بنموذج RGB .
- هناك مكون آخر هو Alpha الذي يحدد درجة الشفافية إذا كانت Alpha تساوي الصفر يكون الشكل شفافاً بينما إذا كانت 255 يكون معتماً (غير شفاف) ويكون النموذج اللوني في هذه الحالة هو ARGB .
- يحدد الرقم الأول في قيمة RGB مقدار اللون الأحمر في اللون ، بينما يحدد الثاني مقدار اللون الأخضر ويحدد الثالث مقدار اللون الأزرق. كلما كبرت القيمة ، زاد مقدار ذلك اللون المعين .

- الصيغة المستخدمة لإنشاء غرض من الصف Color هي :

```
Color   color  = new Color ( r , g , b ) ;
```

- حيث تحدد البارامترات r , g , b اللون بواسطة عناصره الثلاث (الأحمر ، الأخضر ، الأزرق) ، على سبيل المثال :

```
Color   color  = new Color ( 128 , 100 , 100 ) ;
```

- يمكن تحديد اللون بطريقة ثانية وذلك باستخدام واحد من الألوان القياسية الموضحة في الجدول(1) .

Constants in structure Color (all are public static)	RGB value	Constants in structure Color (all are public static)	RGB value
Orange	255, 200, 0	White	255, 255, 255
Pink	255, 175, 175	Gray	128, 128, 128
Cyan	0, 255, 255	DarkGray	64, 64, 64
Magenta	255, 0, 255	Red	255, 0, 0
Yellow	255, 255, 0	Green	0, 255, 0
Black	0, 0, 0	Blue	0, 0, 255

الجدول(1) Color structure static constants and their RGB values.

- تُرسم السلاسل strings والأشكال الهندسية باستخدام الفرشاة Brushes والأقلام .Pens.
- يُستخدم القلم Pen، الذي يعمل بطريقة مشابهة للقلم العادي ، لرسم الخطوط ، وتتطلب معظم توابع الرسم غرضاً من الصف Pen.
- تقبل بواني الصف Pen التحميل الزائد من أجل تحديد ألوان وعرض الخطوط لرسم الشكل المطلوب ، حيث يوفر فضاء الحالة System.Drawing أيضاً مجموعة أقلام تحتوي على أقلام محددة مسبقاً.
- نحدد لون خط الرسم كما يلي :

```
Pen pen = new Pen(Color.AliceBlue)
```

- تغيير لون خط الرسم كما يلي :

```
pen.Color = Color.Red;
```

- كل الصفوف المشتقة من الصف المجرد Brush تحدد الأغراض التي تلون الأجزاء الداخلية للأشكال الرسومية graphical shapes (على سبيل المثال ، يأخذ باني الصف SolidBrush غرض من الصف Color من أجل تلوين الرسم) .
- في معظم توابع التعبئة Fill ، تلون الفرشاة الشكل بلون أو نموذج أو صورة ، ويلخص الجدول (2) مختلف أنواع الفرشاة ووظائفها.
- نحدد لون تعبئة شكل ما ، كما يلي :

```
SolidBrush brush = new SolidBrush(Color.Blue);
```

- تغيير لون تعبئة شكل ، كما يلي :

```
brush.Color = Color.Red
```


Class	Description
HatchBrush	Uses a rectangular brush to fill a region with a pattern. The pattern is defined by a member of the **HatchStyle** enumeration, a foreground color (with which the pattern is drawn) and a background color.
LinearGradient-Brush	Fills a region with a gradual blend of one color into another. Linear gradients are defined along a line. They can be specified by the two colors, the angle of the gradient and either the width of a rectangle or two points.
SolidBrush	Fills a region with one color. Defined by a **Color** object.
TextureBrush	Fills a region by repeating a specified **Image** across the surface.

الجدول(2) Classes that derive from class **Brush**.

الصف Font
Font Class

- يمكن تحديد الخطوط من أجل العناصر أو الأغراض التي نرسمها . كما يمكن استخدام قياسات الخط من أجل حساب حجم الخط .
- تم تغليف الخط وقياساته ضمن الصف Font .
- نحتاج من أجل تحديد الخط إلى إنشاء غرض من الصف Font كما يلي :

```
Font myfont = new Font( Name, Size,  Style );
```

- على سبيل المثال :

```
Font font = new Font(" Tahoma ", 16, FontStyle.Bold);
```


- أو يمكن إنشاء غرض من الصف FontStyle كما يلي :

```
FontStyle  style  = FontStyle.Bold;
style = FontStyle.Regular;
Font  timesNewRoman = new  Font  ( "Times New Roman", 12, style );
```

- **يحتوي الصف Font على عدد من البواني** ويتطلب **اسم الخط *font name*** كوسيط أول، وهو عبارة عن سلسلة تمثل نوع الخط الذي يدعمه النظام حاليًا ، و تتضمن الخطوط الشائعة في Microsoft هو SansSerif و Serif .
- ويتطلب الباني أيضاً **حجم الخط *font size*** كوسيط ثاني **ونمط الخط *font style*** كوسيط ثالث و هو نمط من الأنماط التالية : Bold أو Italic أو Regular أو Strikeout أو Underline ، و يمكن دمج أنماط الخط بواسطة المعامل OR "|" (على سبيل المثال ، FontStyle.Bold | FontStyle.Italic، مما يجعل الخط مائلًا وغامقًا) وأيضاً المعامل Ex-OR .
- **ويبين الجدول(3) خصائص الصف Font .**

Property	Description
Bold	Tests a font for a bold font style. Returns true if the font is bold.
FontFamily	Represents the FontFamily of the Font (a grouping structure to organize fonts and define their similar properties).
Height	Represents the height of the font.
Italic	Tests a font for an italic font style. Returns true if the font is italic.
Name	Represents the font's name as a string.
Size	Returns a float value indicating the current font size measured in design units (design units are any specified units of measurement for the font).
SizeInPoints	Returns a float value indicating the current font size measured in points.
Strikeout	Tests a font for a strikeout font style. Returns true if the font is in strikeout format.
Underline	Tests a font for a underline font style. Returns true if the font is underlined.

الجدول(3) Font class read-only properties.

رسم نص (سلسلة محرفية)
Draw String

- يمكن استخدام توابع الرسم المختلفة والمعرفة في الصف Graphics من أجل رسم النصوص والأشكال الهندسية .
- نستخدم التابع DrawString من أجل رسم نص ويحتوي على عدد من الوسائط وله الشكل التالي :

```
DrawString(string str,Font font, Brush brush, Point point);
```

- str : هي السلسلة الواجب رسمها .
- font : هو غرض من الصف Font ، الذي له الشكل التالي :

```
Font font = new Font(" Tahoma ", 16, FontStyle.Bold);
```

- brush : هو غرض من الصف Brush الذي يحدد لون رسم السلسلة .
- point : هو غرض من الصف Point الذي يحدد بداية رسم السلسلة .

- حيث يحتوي الصف Font معلومات عن الخط المستخدم لعرض السلسلة المحرفية ، ويمتلك خاصية اسم الخط وحجمه ، كما يمتلك أيضاً خاصية جعل الخط عريض Bold أو Italic أو وضع تسطير تحت الخط .
- يتم استدعاء تابع رسم السلسلة كما يلي :

```
this.CreateGraphics().DrawString( .. , .. ,..,.. ) ;
```

- يبين الشكل التالي رسم السلسلة المحرفية التالية :

" Welcome To C#   Programming ! "

- في المثال التالي يتم أنشاء :

1. زر Button نكتب علية العبارة String .
2. مربع نص نكتب من خلاله الرسالة الواجب رسمها على الـ Form .
3. زر Button آخر نكتب عليه Clear من أجل مسح الرسم على النافذة Form .

مثال(1) : رسم نص

البرنامج

```
using System;
using System.Collections.Generic;

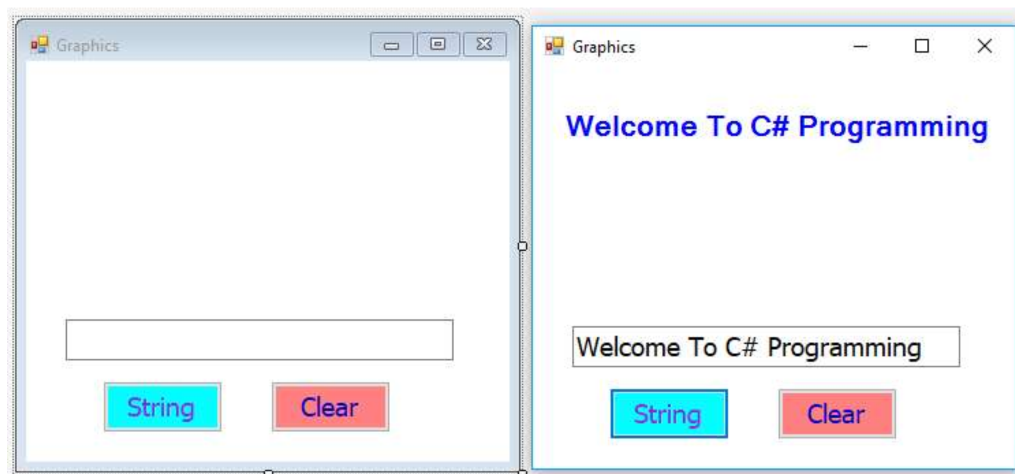
using System.Drawing;

using System.Windows.Forms;

namespace WindowsFormsApplication24
{
    public partial class Graphics : Form
    {
        public Graphics()
        {
            InitializeComponent();
        }
```

```
private void btnString_Click(object sender, EventArgs e)
    {
      string str = txtString.Text;
      Font myFont = new Font(" Tahoma ", 16, FontStyle.Bold);
      Point point = new Point(20, 30);
     this.CreateGraphics().DrawString(str, myFont, Brushes.Blue, point);
   }

 private void btncClear_Click(object sender, EventArgs e)
        {
            this.CreateGraphics().Clear(Color.White);
        }
   }//end class
}// end namespace
```

تمرين (1)

- اكتب برنامجاً بلغة C# يحتوي على نافذة اسمها Graphics بحيث يتم رسم سلسلتين ضمن إطار كما هو موضح بالشكل التالي :



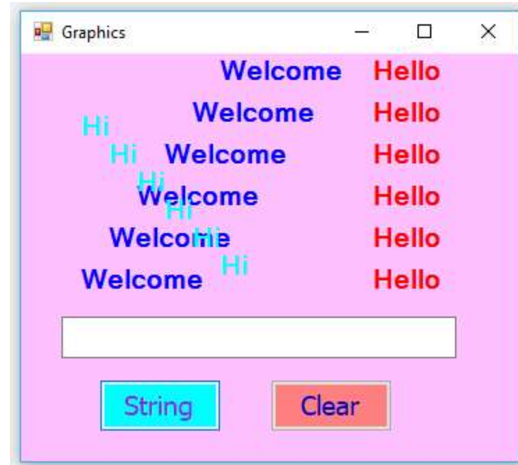
تمرين (2)

- اكتب برنامجاً بلغة C# يحتوي على نافذة اسمها Graphics بحيث يتم رسم السلسلة التالية سبع مرات ضمن إطار باستخدام حلقة for كما هو موضح بالشكل التالي :



تمرين (3)

- اكتب برنامجاً بلغة C# يحتوي على نافذة اسمها Graphics بحيث يتم رسم السلسلة التالية ست مرات ضمن إطار باستخدام حلقة for كما هو موضح بالشكل التالي :
-

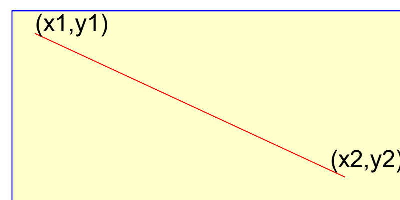


رسم الخطوط
Draw Lines

- يتم رسم الخطوط باستخدام التابع DrawLine المعرف في الصف Graphics ،حيث يتلقى التابع DrawLine غرضاً من الصف Pen وغرضين من الصف Point يستخدمان كنقطتي بداية ونهاية للخط .
- يمكن استخدام التابع التالي من أجل رسم خط مستقيم :

DrawLine(**Pen p** , **int** x1, **int** y1, **int** x2, **int** y2)

تمثل المتحولات x1,y1,x2,y2 نقطة البداية (x1,y1) ونقطة النهاية (x2,y2) ، أما p يحدد لون خط الرسم .



- يتم عادة وضع الـ code اللازم للرسم عادة ضمن الحدث Paint للنافذة Form .
- **المثال التالي يرسم خط أحمر :**

```
DrawLine(Pens.Red, 2 0, 20, 250, 20);
```

- **أو يمكن رسم المستقيم أو الخط بإنشاء غرضين من الصف Point كما يلي :**

```
Point p1 = new Point(20, 20);
Point p2 = new Point(250, 20);
DrawLine(Pens.Red, p1,p2);
```

- **ملاحظة : لرسم خط عريض كما يلي :**

```
Pen pp = new Pen( Color.Red, 5 );
```

حيث الوسيط الثاني عرض الخط .

- **يبين البرنامج التالي معالج الحدث Paint للنافذة Form والذي يتم فيه رسم 10 خطوط بلون أحمر .**

مثال(2) : رسم خطوط

البرنامج

```
using System;

using System.Drawing;

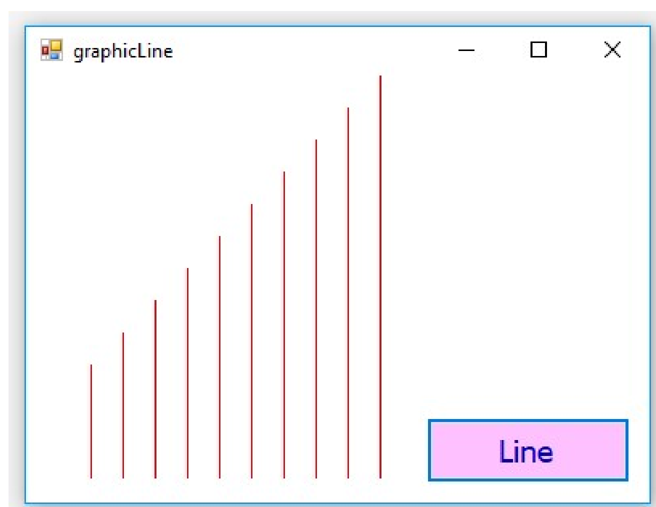
using System.Windows.Forms;

namespace graphic1
{
    public partial class LineForm1 : Form
    {
        public LineForm1()
        {
            InitializeComponent();
        }
```

```
        private void btnLine_Click(object sender, EventArgs e)
        {
            //this.CreateGraphics().DrawLine(Pens.Blue, 20, 20, 250, 20);
            Point start = new Point(20, 250);
            Point stop = new Point(20, 200);
            // drae Lines
            for(int i=0; i<10 ; i++ )
            {
                start.X = start.X +20  ;
                stop.X = stop.X + 20;
                stop.Y = stop.Y - 20;

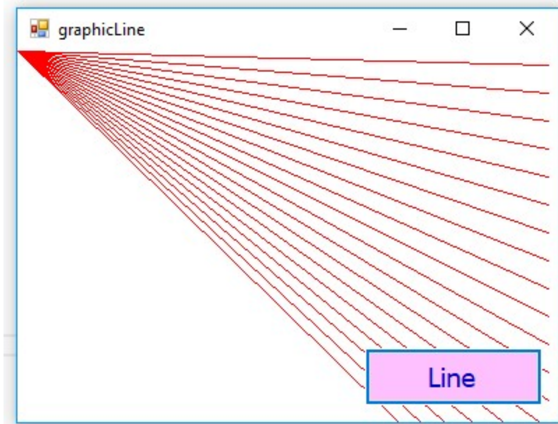
              this.CreateGraphics().DrawLine(Pens.Red, start,stop);

            }// end for
        }// end btn
    }// end class
} // end namespace
```

تمرين(1)

- اكتب برنامجاً بلغة #C لرسم الشكل الهندسي التالي ضمن النافذة Form (رسم 20خط مستقيم) باستخدام حلقة for .



حل التمرين(1)

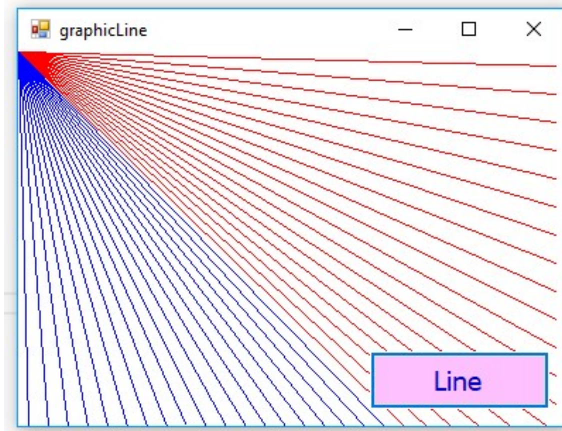
البرنامج

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace graphic1
{
    public partial class LineForm1 : Form
    {
        public LineForm1()
        {
            InitializeComponent();
        }
       private void btnLine_Click(object sender, EventArgs e)
        {
            for (int i = 1; i <= 20; i++)
            {
              this.CreateGraphics().DrawLine(Pens.Red,0, 0, 380, 390 - 20 * i);
            }// end for
        }// end btn
    }// end class
} // end namespace
```

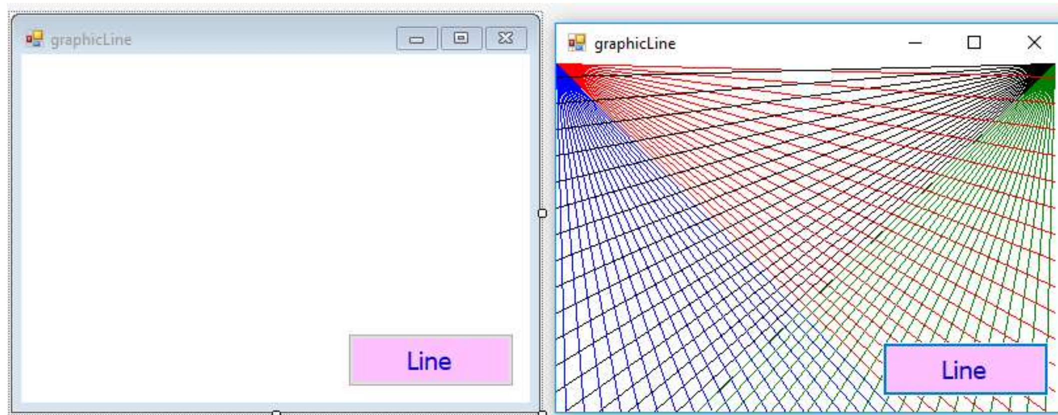
تمرين (2)

• اكتب برنامجاً بلغة C# لرسم الشكل الهندسي التالي ضمن النافذة Form (رسم 20خط مستقيم أحمر و20 خط أزرق) باستخدام حلقة for ، كما هو مبين بالشكل التالي .



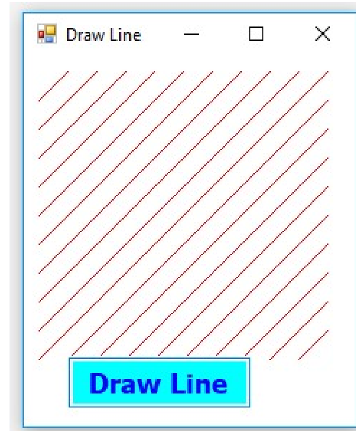
تمرين (3)

• اكتب برنامجاً بلغة C# لرسم الشكل الهندسي التالي ضمن النافذة Form (رسم 20خط مستقيم أحمر و20 خط أزرق و20 خط أخطر و20 خط أسود) باستخدام حلقة for ، كما هو مبين بالشكل التالي .



تمرين (4)

- اكتب برنامجاً بلغة C# من أجل رسم الأشكال الهندسية التالية ضمن النافذة Form .

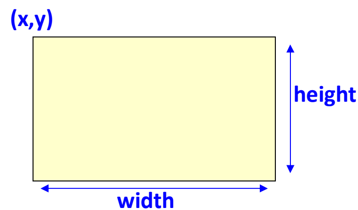


رسم مستطيل
Draw Rectangle

- يقدم الصف Graphics تابعين من أجل رسم المستطيلات الفارغة والممتلئة .
- يمكن استخدام التابع التالي من أجل رسم مستطيل منبسط :

```
DrawRectangle( Pen p , int x, int y, int width, int height)
```

حيث يمثل المتحولان (x,y) الزاوية اليسارية العليا للمستطيل بينما يمثل المتحولان (width,height)عرض وطول المستطيل ، أما p فيمثل غرض من الصف Pen الذي يمثل لون خط الرسم .



- التابع التالي يستخدم من أجل رسم مستطيل ممتلئ :

```
FillRectangle(Brush  b, int x, int y, int width, int height )
```

حيث تحدد الفرشاة (Brush) لون التعبئة داخل المستطيل .

- المثال التالي يبين رسم مستطيل منبسط وممتلئ :

```
DrawRectangle(Pens.Blue, 200, 30, 200, 300);
FillRectangle(Brushes.Red, 200, 330, 200, 300);
```

- يبين البرنامج التالي معالج الحدث Paint للنافذة Form والذي يتم فيه رسم مستطيل منبسط بلون أحمر وآخر بلون أزق ومستطيل ممتلئ بلون Cyan وآخر ممتلئ بلون زهري.

مثال(3) : رسم مستطيلات

البرنامج

```
using System;

using System.Drawing;

using System.Windows.Forms;

namespace Rectangle
{
    public partial class RecForm1 : Form
    {
        public LineForm1()
        {
            InitializeComponent();
        }
```

```
private void btnRec_Click(object sender, EventArgs e)
        {

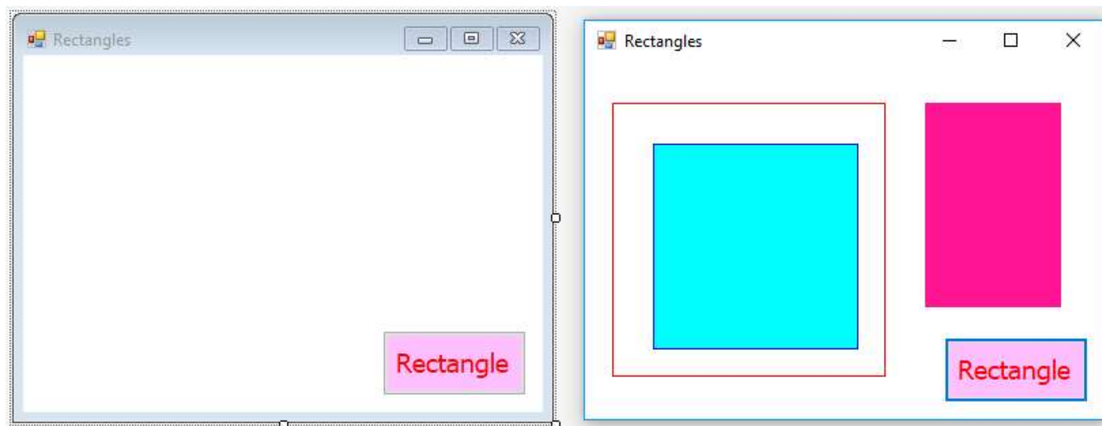
            this.CreateGraphics().DrawRectangle(Pens.Red, 20 , 30, 200, 200);

            this.CreateGraphics().FillRectangle(Brushes.Cyan, 50, 60, 150, 150);

            this.CreateGraphics().DrawRectangle(Pens.Blue, 50, 60, 150, 150);

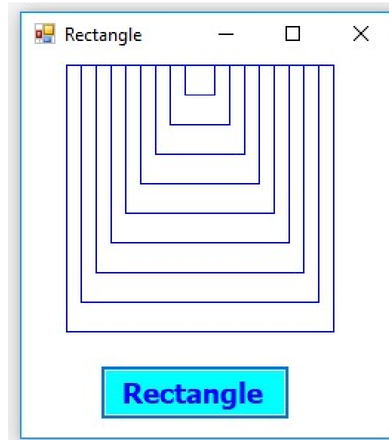
            this.CreateGraphics().FillRectangle(Brushes.DeepPink, 250, 30, 100, 150);

        }//end btn
    } //end class
} //end namespace
```

تمرين (5)

- اكتب برنامجاً بلغة C# من أجل رسم الأشكال الهندسية التالية ضمن النافذة Form .

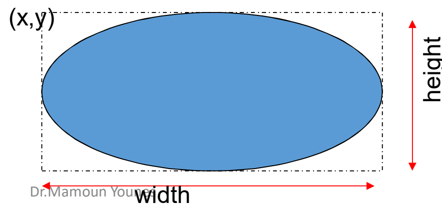


رسم الأشكال البيضاوية
Draw Ellipes

- يمكن رسم شكل بيضاوي (قطع ناقص) في لغة C# بالاعتماد على المستطيل الذي يحيط به ، لذلك نكتب المتحولات في توابع رسم القطع الناقص وكأننا نرسم مستطيلاً .
- **يقدم الصف Graphics تابعين لرسم الشكل البيضاوي (قطع ناقص) .**
- **التابع الأول لرسم أشكال منبسطة (فارغة) وله الشكل التالي :**

DrawEllipse(**Pen p**, **,int** x, **int** y, **int** width, **int** height)

يحدد المتحولان x,y الزاوية اليسارية العليا للمستطيل الذي يحيط بالقطع الناقص بينما يحدد المتحولان width, height عرض وطول المستطيل .



- التابع الثاني يستخدم من أجل رسم قطع ناقص ممتلئ ، وله الشكل التالي :

FillEllipse(Brush b , **int** x, **int** y, **int** width, **int** height)

- المثال التالي يبين رسم قطع ناقص منبسط وممتلئ :

```
DrawEllipse(Pens.Yellow, 200, 30, 200, 300);
FillEllipse(Brushes.Blue,200, 30, 200, 300);
```

- يبين البرنـامج التـالي معالج الحدث **Paint** للنافذة **Form** والـذي يـتم فيـه رسم أشكال بيضاويـة منبسطة وممتلئة ن بحيث يتم رسم شكل بيضاوي ممتلئ ومنبسط بلون أحمر ، ورسم شكل بيضاوي ممتلئ ومنبسط بلون أزرق ، كما هو موضح في البرنـامج التالي .

مثال(4) : رسم اشكال بيضاوية

البرنامج

```
using System;

using System.Drawing;

using System.Windows.Forms;

namespace drawEillpse
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
```

```
private void btnEllipse_Click(object sender, EventArgs e)
  {

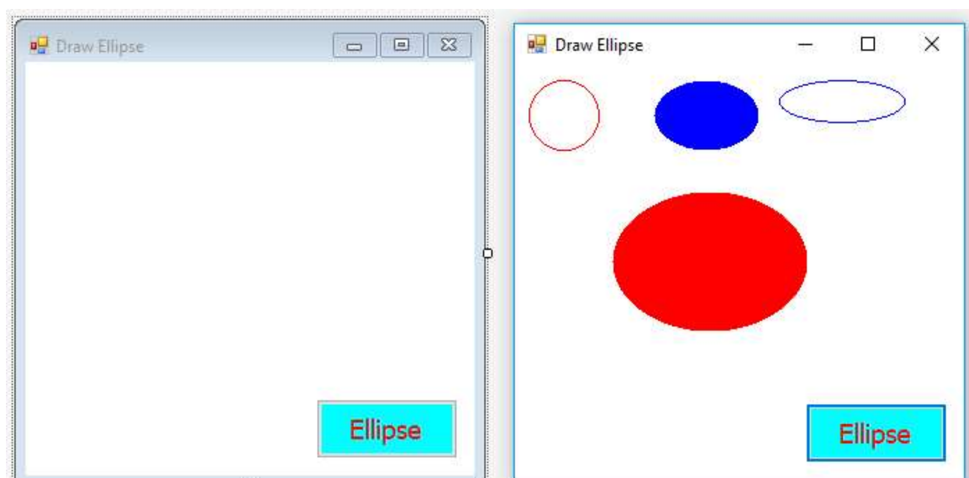
      this.CreateGraphics().DrawEllipse(Pens.Red, 10, 10, 50, 50);

      this.CreateGraphics().FillEllipse(Brushes.Blue, 100, 10, 75, 50);

      this.CreateGraphics().DrawEllipse(Pens.Blue, 190, 10, 90, 30);

      this.CreateGraphics().FillEllipse(Brushes.Red, 70, 90, 140, 100);

  }// end btn
 } //end class
} // end namespace
```

Graphics Drawing Methods and Descriptions.

Note: Many of these methods are overloaded—consult the documentation for a full listing.

`DrawLine( Pen p, int x1, int y1, int x2, int y2 )`
Draws a line from (**x1**, **y1**) to (**x2**, **y2**). The **Pen** determines the color, style and width of the line.

`DrawRectangle( Pen p, int x, int y, int width, int height )`
Draws a rectangle of the specified width and height. The top-left corner of the rectangle is at point (**x**, **y**). The **Pen** determines the color, style, and border width of the rectangle.

`FillRectangle( Brush b, int x, int y, int width, int height )`
Draws a solid rectangle of the specified width and height. The top-left corner of the rectangle is at point (**x**, **y**). The **Brush** determines the fill pattern inside the rectangle.

`DrawEllipse( Pen p, int x, int y, int width, int height )`
Draws an ellipse inside a rectangle. The width and height of the rectangle are as specified, and its top-left corner is at point (**x**, **y**). The **Pen** determines the color, style and border width of the ellipse.

`FillEllipse( Brush b, int x, int y, int width, int height )`
Draws a filled ellipse inside a rectangle. The width and height of the rectangle are as specified, and its top-left corner is at point (**x**, **y**). The **Brush** determines the pattern inside the ellipse.

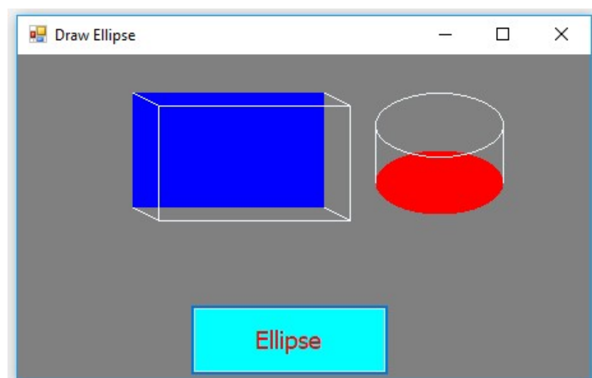
الجدول(4) **Graphics** methods that draw lines, rectangles and ovals.

يوضح الجدول (4) توابع الرسم في الصف Graphics لرسم الخطوط والمستطيلات والأشكال البيضاوية

تمرين (6)

- اكتب برنامجاً بلغة C# لرسم الشكل الهندسي التالي ضمن النافذة Form حيث يتم رسم ما يلي :

1. create filled rectangle
2. draw lines to connect rectangles
3. draw top rectangle
4. draw base Ellipse
5. draw connecting lines
6. draw Ellipse outline



حل التمرين (6)

البرنامج

```
using System;

using System.Drawing;

using System.Windows.Forms;

namespace drawEillpse
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnEllipse_Click(object sender, EventArgs e)
        {
           // create filled rectangle
            this.CreateGraphics().FillRectangle(Brushes.Blue, 90, 30, 150, 90);
```



```
           // draw lines to connect rectangles
            this.CreateGraphics().DrawLine(Pens.AliceBlue, 90, 30, 110, 40);
            this.CreateGraphics().DrawLine(Pens.AliceBlue, 90, 120, 110, 130);
            this.CreateGraphics().DrawLine(Pens.AliceBlue, 240, 30, 260, 40);
            this.CreateGraphics().DrawLine(Pens.AliceBlue, 240, 120, 260, 130);

            // draw top rectangle
             this.CreateGraphics().DrawRectangle( Pens.AliceBlue, 110, 40, 150, 90 );

            // draw base Ellipse
             this.CreateGraphics().FillEllipse( Brushes.Red, 280, 75, 100, 50 );

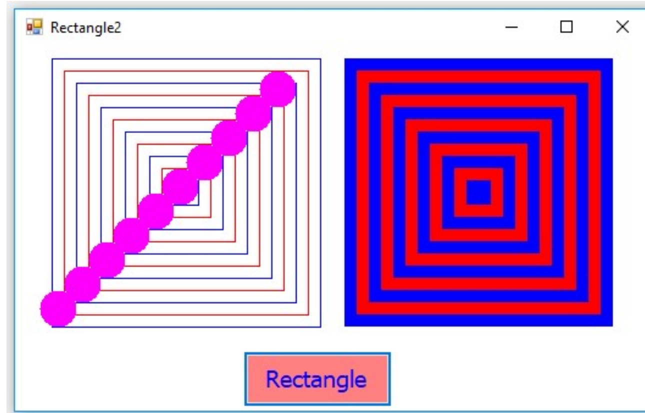
            // draw connecting lines
             this.CreateGraphics().DrawLine( Pens.AliceBlue, 380, 55, 380, 100 );
             this.CreateGraphics().DrawLine( Pens.AliceBlue, 280, 55, 280, 100 );

            // draw Ellipse outline
             this.CreateGraphics().DrawEllipse( Pens.AliceBlue, 280, 30, 100, 50 );

        }// end btn
    } //end class
} // end namespace
```

تمرين (7)

- اكتب برنامجاً بلغة C# لرسم الشكل الهندسي التالي ضمن النافذة Form (رسم 10 مربعات أو مستطيلات حمراء وزرقاء منبسطة وممتلئة بالتناوب باستخدام حلقة for ، كما هو مبين بالشكل التالي :



حل التمرين (7)

```
using System;

using System.Drawing;

using System.Windows.Forms;

namespace RecEx2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnRec_Click(object sender, EventArgs e)
        {
            Pen pen = new Pen(Color.AliceBlue);
     //---------------------------------------------------------------
```

```
for (int i = 0; i <= 10; i++)
 {

  pen.Color = Color.Red;
  if (i % 2 == 0)
  pen .Color = Color.Blue;
 this.CreateGraphics().DrawRectangle(pen, 130 - 10 * i, 110 - 10 * i,
                                                       20 + 20 * i, 20 + 20 * i);
  }//end for
 //----------------------------------------------------------------
  SolidBrush brush = new SolidBrush(Color.Blue);

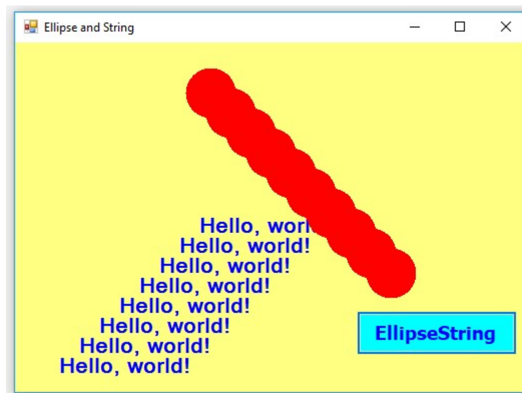
  for (int i = 0; i <= 10; i++)
  {
    brush.Color = Color.Red;
    if (i % 2 == 0)
    brush.Color = Color.Blue;
```



```
//this.CreateGraphics().FillRectangle(brush, 370 - 10 * i, 110 - 10 * i,
                                                       20 + 20 * i, 20 + 20 * i);
  this.CreateGraphics().FillRectangle(brush, 270 + 10 * i, 10 + 10 * i,
                                                      220 - 20 * i, 220 - 20 * i);
   }//end for
//---------------------------------------------------------------
  brush.Color = Color.Magenta;
  for (int i = 1; i <= 10; i++)
   {
  this.CreateGraphics().FillEllipse(brush, 20 * (10 + 1 - i), 20 * i, 30, 30);
   }// end for
        }// end btn
    }//end class
}// end namespace
```

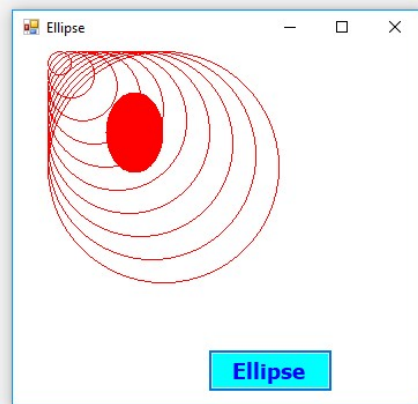
تمرين (8)

- ارسم السلسلة !Hello, world ثمان مرات باللون الأزرق وباستخدام حلقة for وذلك من خلال تغير الإحداثيات (x,y) اعتبارا من النقطة (150,200).
- ارسم قطع ناقص ممتلئ لونه RED(عشر مرات) حيث إحداثيات الزاوية اليسارية العليا هي (100, 5) وطول وعرض المستطيل height = 50 , width=50. وذلك من خلال تغير الإحداثيات (x,y) اعتبارا من النقطة (100,5).
- بحيث نحصل على الشكل التالي :



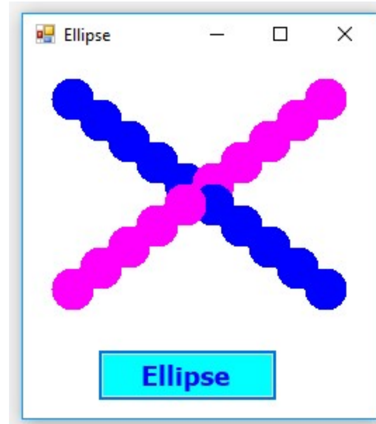
تمرين (9)

- ارسم قطع ناقص منبسط لونه RED(عشرة مرات) حيث إحداثيات الزاوية اليسارية العليا هي (100, 5) وطول وعرض المستطيل height = 50 , width=50. وذلك من خلال تغير الإحداثيات (x,y) اعتبارا من النقطة (100,5).
- ارسم قطع ناقص ممتلئ لونه RED حيث إحداثيات الزاوية اليسارية العليا هي (80, 40) وطول وعرض المستطيل height = 70 , width=50.



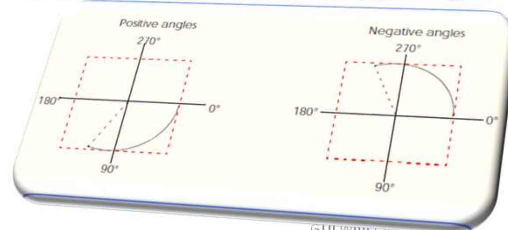
تمرين (10)

- اكتب برنامجاً بلغة C# من أجل رسم الأشكال الهندسية التالية ضمن النافذة Form .



رسم الأقواس
Drawing Arcs and Pie

- يعتبر القوس جزء من القطع الناقص (الشكل البيضاوي) لذلك يتم رسم القوس بالاعتماد على المستطيل المحيط بالشكل البيضاوي .
- تبدأ من زاوية البداية وتستمر لعدد محدد من الدرجات يسمى زاوية القوس، ويقال إن القوس يمسح بزاوية قوسه ، بداية من زاوية البدء.
- يتم قياس الأقواس التي تدور باتجاه عقارب الساعة بدرجات موجبة ، في حين يتم قياس الأقواس التي تمسح في اتجاه عكس عقارب الساعة بالدرجات السالبة.



- توابع رسم الأقواس في الصف Graphics هي :

DrawArc , DrawPie and FillPie .

يُستخدم التابعان التاليان من أجل رسم قوس فارغ :

```
DrawArc( Pen p, int x, int y, int width, int height, int startAngle, int sweepAngle )
DrawPie( Pen p, int x, int y, int width, int height, int startAngle, int sweepAngle )
```

- أما التابع التالي يُستخدم من أجل رسم قوس ممتلئ :

```
FillPie(Brush p, int x, int y, int width, int height, int startAngle, int sweepAngle )
```

- يحدد المتحولان x,y الزاوية اليسارية العليا للمستطيل الذي يحيط بالقطع الناقص بينما يحدد المتحولان width, height عرض وطول المستطيل .
- **ويبين الجدول (3) توابع رسم الأقواس في الصف Graphics .**

Graphics Methods And Descriptions
Note: Many of these methods are overloaded—consult the documentation for a complete listing.
`DrawArc( Pen p, int x, int y, int width, int height, int startAngle, int sweepAngle )` Draws an arc of an ellipse, beginning from angle **startAngle** (in degrees) and sweeping **sweepAngle** degrees. The ellipse is defined by a bounding rectangle of width **w**, height **h** and upper-left corner (**x**,**y**). The **Pen** determines the color, border width and style of the arc.
`DrawPie( Pen p, int x, int y, int width, int height, int startAngle, int sweepAngle )` Draws a pie section of an ellipse, beginning from angle **startAngle** (in degrees) and sweeping **sweepAngle** degrees. The ellipse is defined by a bounding rectangle of width **w**, height **h** and upper-left corner (**x**,**y**). The **Pen** determines the color, border width and style of the arc.
`FillPie( Brush b, int x, int y, int width, int height, int startAngle, int sweepAngle )` Functions similarly to **DrawPie**, except draws a solid arc (i.e., a sector). The **Brush** determines the fill pattern for the solid arc.

الجدول(4) **Graphics** methods for drawing arcs.

مثال (5) :رسم الأقواس

- يقوم المثال التالي بـ :
  - إنشاء زر Button نكتب عليه Arc لرسم قوس حيث نقطة البداية (20,40) ، عرض وطول المستطيل 200 ، وزاوية البدء 0 ويمسح 360 درجة ، ورسم قوس آخر لـه نفس نقطة البداية حيث زاوية البدء 180ويمسح زاوية قدرها 90 درجة .
  - إنشاء زر Button ثاني نكتب عليه Pie لرسم قطاع من شكل بيضاوي حيث نقطة البداية (200,40) ، وعرض وطول المستطيل 200 ، وزاوية البدء 0 ويمسح 90 درجة ، ورسم قوس آخر له نفس نقطة البداية حيث زاوية البدء -90 ويمسح زاوية قدرها -90 درجة .
  - إنشاء زر Button ثالث نكتب عليه Clear يقوم بمسح المحتوى الرسومي الموجود على الشاشة .
  - إنشاء عنصر التحكم بالأرقام NumericUpDown يقوم برسم عدد كبير من القطاعات Pie .

البرنامج

```
using System;

using System.Drawing;

using System.Windows.Forms;

namespace Arcs1
{
    public partial class ArcForm : Form
    {
        public ArcForm()
        {
            InitializeComponent();
        }

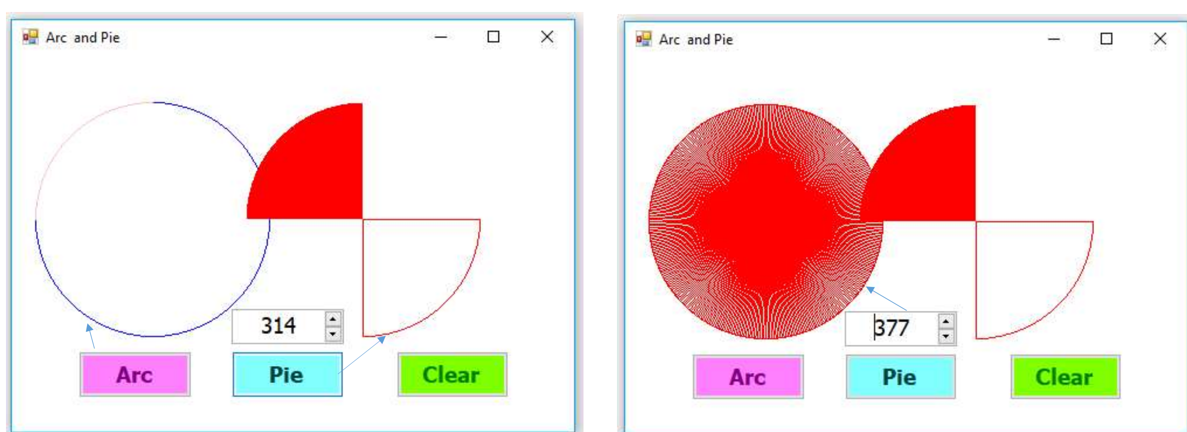
        private void btn1_Click(object sender, EventArgs e)
        {
            this.CreateGraphics().DrawArc(Pens.Blue, 20, 40, 200, 200, 0, 360);
            this.CreateGraphics().DrawArc(Pens.Pink, 20, 40, 200, 200, 180, 90);
        }
```

```
private void btnPie_Click(object sender, EventArgs e)
  {
      this.CreateGraphics().DrawPie(Pens.Red, 200, 40, 200, 200, 0, 90);
      this.CreateGraphics().FillPie(Brushes.Red, 200, 40, 200, 200, -90, -90);
  }

  private void btnClear_Click(object sender, EventArgs e)
  {
      this.CreateGraphics().Clear(Color.White);
  }

 private void numericUpDown1_ValueChanged(object sender, EventArgs e)
  {
    this.CreateGraphics().DrawPie(Pens.Red, 20, 40, 200, 200, 0,
                                                         (float)numericUpDown1.Value);

    }//end class
}// end namespace
```

مثال (6) :رسم الأقواس والمستطيلات

- يقوم المثال التالي بــ :

➢ إنشاء زر Button نكتب عليه (Arc 0,360) لرسم قوس حيث نقطة البداية (15,35) ، عرض وطول المستطيل 80 ، وزاوية البدء 0 ويمسح 360 درجة ، ورسم مستطيل له نفس نقطة البداية وطول وعرض المستطيل .

➢ إنشاء زر Button نكتب عليه (Arc 0,110) لرسم قوس حيث نقطة البداية (100,35) ، عرض وطول المستطيل 80 ، وزاوية البدء 270 ويمسح 90- درجة ، ورسم مستطيل له نفس نقطة البداية وطول وعرض المستطيل .

➢ إنشاء زر Button نكتب عليه (Arc 0,-270) لرسم قوس حيث نقطة البداية (185,35) ، عرض وطول المستطيل 80 ، وزاوية البدء 0 ويمسح 270- درجة ، ورسم مستطيل له نفس نقطة البداية وطول وعرض المستطيل .

➢ إنشاء زر Button نكتب عليه (FillPie 0,360) لرسم قطاع ممتلئ حيث نقطة البداية (15,35) ، عرض وطول المستطيل 80 ، وزاوية البدء 0 ويمسح 360 درجة.

➢ إنشاء زر Button نكتب عليه (FillPie 270,-90) لرسم قطاع ممتلئ حيث نقطة البداية (100,35) ، عرض وطول المستطيل 80 ، وزاوية البدء 270 ويمسح 90- درجة.

➢ إنشاء زر Button نكتب عليه (FillPie 0,-270) لرسم قطاع ممتلئ حيث نقطة البداية (185,35) ، عرض وطول المستطيل 80 ، وزاوية البدء 0 ويمسح 270- درجة.

➢ إنشاء زر Button ثالث نكتب عليه Clear يقوم بمسح المحتوى الرسومي الموجود على الشاشة .

- البرنامج التالي يوضح هذا التصميم .

البرنامج

```
using System;

using System.Drawing;

using System.Windows.Forms;

namespace Arcs2
{
    public partial class ArcsForm1 : Form
    {
        public ArcsForm1()
        {
            InitializeComponent();
        }

        private void btn2_Click(object sender, EventArgs e)
        {
            // start at 0 and sweep 110 degrees
            this.CreateGraphics().DrawRectangle(Pens.Red, 100, 35, 80, 80);
            this.CreateGraphics().DrawArc(Pens.Blue, 100, 35, 80, 80, 0, 110);
        }
```



```
       private void btn1_Click(object sender, EventArgs e)
        {
            // start at 0 and sweep 360 degrees
            this.CreateGraphics().DrawRectangle(Pens.Red, 15, 35, 80, 80);
            this.CreateGraphics().DrawArc(Pens.Blue, 15, 35, 80, 80, 0, 360);
        }

        private void Arc3_Click(object sender, EventArgs e)
        {
            // start at 0 and sweep -270 degrees
            this.CreateGraphics().DrawRectangle(Pens.Red, 185, 35, 80, 80);
            this.CreateGraphics().DrawArc(Pens.Blue, 185, 35, 80, 80, 0, -270);

        }

        private void btn4_Click(object sender, EventArgs e)
        {
            // start at 0 and sweep 360 degrees
            this.CreateGraphics().FillPie(Brushes.Blue, 15, 120, 80, 40, 0, 360);
        }
```

```
        private void btn5_Click(object sender, EventArgs e)
          {
              // start at 270 and sweep -90 degrees
              this.CreateGraphics().FillPie(Brushes.Blue, 100, 120, 80, 40, 270, -90);
          }

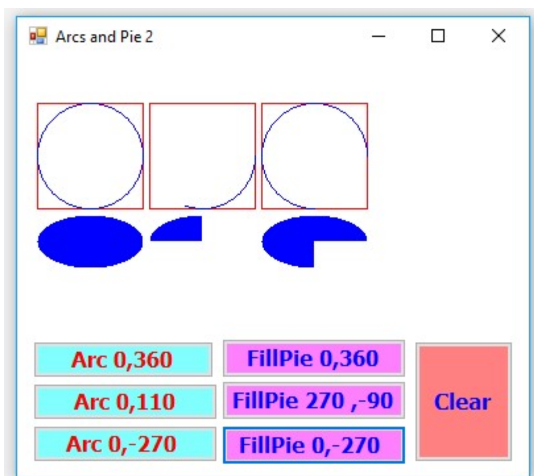
          private void button6_Click(object sender, EventArgs e)
          {
              // start at 0 and sweep -270 degrees

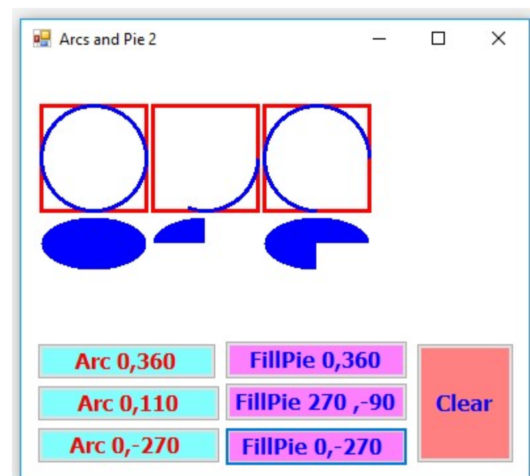
              this.CreateGraphics().FillPie(Brushes.Blue, 185, 120, 80, 40, 0, -270);

          }

          private void btnClear_Click(object sender, EventArgs e)
          {
              this.CreateGraphics().Clear(Color.White);

          }
      }//end class
   }// end namespace
```



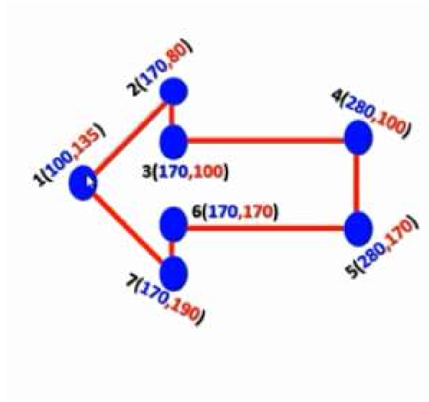
رسم المضلعات والخطوط المتعددة
Drawing Polygons and Polylines

- يغلف الصف Polygon وصفاً كاملاً لمنطقة مغلقة وثنائية البعد ضمن الاحداثيات الفضائية , وتحدد المنطقة بعدد معين من القطع المستقيمة حيث تمثل كل قطعة مستقيمة أحد الجوانب أو الحواف edge التابعة للمضلع .
- يتألف المضلع داخلياً من قائمة من الإحداثيات الثنائية (x,y) والتي تعرف كل منها ذروة تابعة للمضلع , كما يمثل كل إحداثيتان متعاقبتان نقطتي البداية والنهاية لخط يشكل أحد أضلاع المضلع , وتتصل النقطة الأولى مع النقطة الأخيرة بواسطة القطعة المستقيمة التي تغلق المضلع .
- حيث DrawLines يرسم سلسلة من النقاط المتصلة ،بينما DrawPolygon يرسم مضلع مغلق أما FillPolygon يرسم مضلع ممتلئ .
- ويبين الجدول(5) توابع رسم المضلعات في الصف Graphics .

Method	Description
DrawLines	Draws a series of connected lines. The coordinates of each point are specified in an array of Points. If the last point is different from the first point, the figure is not closed.
DrawPolygon	Draws a polygon. The coordinates of each point are specified in an array of Point objects. This method draws a closed polygon, even if the last point is different from the first point.
FillPolygon	Draws a solid polygon. The coordinates of each point are specified in an array of Points. This method draws a closed polygon, even if the last point is different from the first point.

الجدول(4) Graphics methods for drawing polygons.

- يُستخدم البانيين التاليين من أجل صنع غرض من الصف Polygon :
  - الباني DrawPolygon() public يبني مضلعاً فارغاً .
  - الباني DrawPolygon(Pen pen, Point[ ] points) public رسم مضلع غير مصمت .
  - الباني FillPolygon(Brush brush, Point[ ] points) public رسم مضلع ممتلئ أي مصمت .
- البرنامج التالي يقوم برسم :
  - مضلع على شكل سهم ممتلئ وغير ممتلئ .
  - مثلث غير ممتلئ .
  - رسم مضلع سداسي الشكل .



مثال (7) :رسم المضلعات

- يقوم المثال التالي بـ :
  - ➢ إنشاء زر Button نكتب عليه Polygon1 لرسم مضلع على شكل سهم ممتلئ وسهم غير ممتلئ .
  - ➢ إنشاء زر Button نكتب عليه Polygon2 لرسم مضلع على شكل مثلث ورسم مضلع آخر سداسي الشكل .
  - ➢ إنشاء زر Button ثالث نكتب عليه Clear يقوم بمسح المحتوى الرسومي الموجود على الشاشة .
- البرنامج التالي يوضح هذا التصميم .

البرنامج

```
using System;

using System.Drawing;

using System.Windows.Forms;

namespace Polygon1
{
    public partial class PolygonForm : Form
    {
        public PolygonForm()
        {
            InitializeComponent();
        }
        //-----------رسم مضلع على شكل سهم ممتلئ-----------
        private void btnPolygon_Click(object sender, EventArgs e)
        {
            Point p1 = new Point(100, 135);
            Point p2 = new Point(170, 80);
            Point p3 = new Point(170, 100);
            Point p4 = new Point(280, 100);
            Point p5 = new Point(280, 170);
```



```
            Point p6 = new Point(170, 170);
            Point p7 = new Point(170, 190);
            Point[] allp = { p1, p2, p3, p4, p5, p6, p7 };

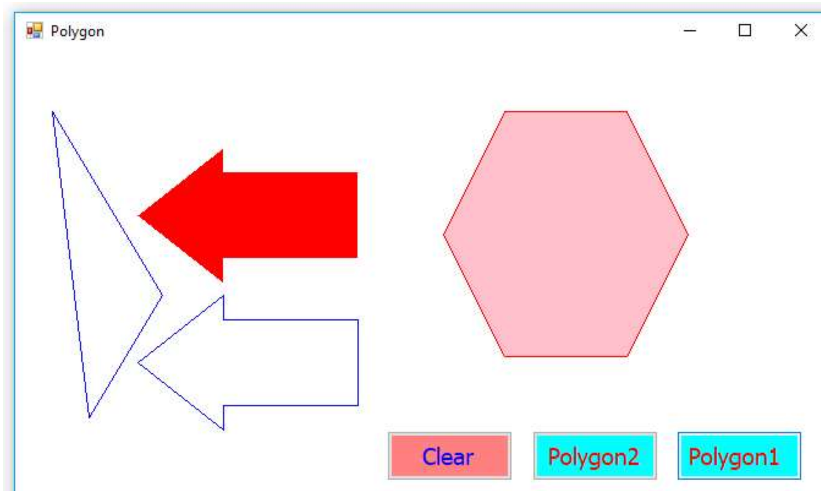
            this.CreateGraphics().FillPolygon(Brushes.Red, allp);
            //-----------رسم مضلع على شكل سهم غير ممتلئ-----------
            Point d1 = new Point(100, 255);
            Point d2 = new Point(170, 200);
            Point d3 = new Point(170, 220);
            Point d4 = new Point(280, 220);
            Point d5 = new Point(280, 290);
            Point d6 = new Point(170, 290);
            Point d7 = new Point(170, 310);
            Point[] d = { d1, d2, d3, d4, d5, d6, d7 };

            this.CreateGraphics().DrawPolygon(Pens.Blue,d);
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            this.CreateGraphics().Clear(Color.White);
        }
```

```
//-----------رسم مضلع على شكل مثلث غير ممتلئ-----------
 private void btnPolygon2_Click(object sender, EventArgs e)
   {
       Point a1 = new Point(30, 50);
       Point a2 = new Point(120, 200);
       Point a3 = new Point(60, 300);
       Point[] a = { a1, a2, a3 };
       this.CreateGraphics().DrawPolygon(Pens.Blue, a);
   //-----------رسم مضلع سداسي الشكل  ممتلئ-----------
       Point b1 = new Point(400, 50);
       Point b2 = new Point(350, 150);
       Point b3 = new Point(400, 250);
       Point b4 = new Point(500, 250);
       Point b5 = new Point(550, 150);
       Point b6 = new Point(500, 50);

       Point[] b = { b1, b2, b3, b4, b5, b6 };
       this.CreateGraphics().FillPolygon(Brushes.Pink, b);
       this.CreateGraphics().DrawPolygon(Pens.Red, b);
   }
 }
}
```

التحكم بمظهر الشكل المراد رسمه
Drawing Figures

- **نبين في هذه الفقرة بعض الإمكانيات المختلفة في التحكم بمظهر الرسم :**
  - كشكل خط الرسم
  - نوع التعبئة أو التظليل .... الخ
- التسلسل الهرمي للفرشاة Brush على سبيل المثال يتضمن :

HatchBrush, LinearGradientBrush, PathGradientBrush and TextureBrush.

- هذه التقنيات موجودة في الصف Graphics2D الذي يرث الصف Graphics .
- يزودنا الصف Graphics2D بعدد من التوابع بعضها يتحكم بشكل خط الرسم ومنها ما يتحكم بأسلوب تعبئة الشكل المرسوم .
- **يوضح البرنامج التالي العديد من ميزات الرسومات ، مثل الخطوط المتقطعة والخطوط السميكة والقدرة على ملء الأشكال باستخدام الأنماط، وهي تمثل فقط بعض القدرات الإضافية من فضاء الحالة System.Drawing .**

حيث :

- **HatchBrush:** يستخدم فرشاة مستطيلة لملء منطقة النموذج pattern ، ويتم تعريف النموذج pattern بواسطة العضو الثابت HatchStyle، لون الواجهة الأمامية(الذي يتم رسم النموذج عليه) ولون الخلفية.
- **LinearGradientBrush :** يملأ منطقة ذات مزيج تدريجي من لون إلى لون آخر، ويتم تعريف التدرجات الخطية على طول خط ، ويمكن تحديدها من خلال لونين ، زاوية التدرج وإما عرض مستطيل أو نقطتين.
- **SolidBrush :** يملأ منطقة ذات لون واحد ، ومعرف بواسطة غرض لوني Color object.
- **TextureBrush :** يملأ منطقة بتكرار صورة محددة عبر السطح.

```
LinearGradientBrush linearBrush = new
LinearGradientBrush(drawArea1, Color.Red,Color.Yellow,
                        LinearGradientMode.ForwardDiagonal);
```

- **الوسيط الأول** هو غرض من الصف Rectangle لتلوين منطقة من الشكل الرسومي وله الشكل التالي :

```
Rectangle drawArea1 = new Rectangle( 5, 35, 10, 10 );
```

- **الوسيط الثاني والثالث** هو التدرج اللوني حيث يبدأ من اللون الأحمر وينتهي باللون الأصفر .
- **الوسيط الرابع** هو ثابت من النوع LinearGradientMode يحدد اتجاه التدرج الخطي في مثالنا يبدأ من الزاوية اليسارية العليا إلى الزاوية اليمينية السفلى كما يلي :

```
LinearGradientMode.ForwardDiagonal
```


مثال على التحكم بمظهر الشكل المراد رسمه

يقوم البرنامج بـ :

- **إنشاء زر اسمه btn** وكتابة عليه Drawing2D .
- **رسم قطع ناقص** يتدرج لونه من اللون الأزرق إلى اللون الأصفر.
- **رسم مستطيل** فارغ عريض لونه أحمر .
- **رسم مستطيل** ممتلئ يتدرج لونه من اللون الأحمر إلى للون الأصفر .
- **رسم مستطيل** ممتلئ يتدرج لونه من اللون الأزرق إلى اللون الأصفر.
- **رسم مستقيم** عريض لونه أخضر .
- **رسم مستقيم** متقطع لونه أصفر .
- **رسم قوس فارغ** يبدأ من الزاوية 0 ويمسح زاوية 270 ولونه Pink .

البرنامج

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Drawing.Drawing2D;

namespace Fdrawig2D
{
    public partial class Drawing2D : Form
    {
        public Drawing2D()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            SolidBrush solidColorBrush = new SolidBrush( Color.Red );
            Pen coloredPen = new Pen( solidColorBrush );
            // draw pie-shaped arc in white
           coloredPen.Color = Color.PaleVioletRed;
           coloredPen.Width = 6;
          this.CreateGraphics().DrawPie( coloredPen, 240, 30, 75, 100, 0, 270 );
```



```
      // draw lines in green and yellow
        coloredPen.Color = Color.Green;
        coloredPen.Width = 10;
         this.CreateGraphics().DrawLine( coloredPen, 395, 30, 320, 150 );
         //-----------------------------------------------------------
        coloredPen.Color = Color.Yellow;
        coloredPen.DashCap = ( DashCap )LineCap.Round;
       coloredPen.DashStyle = DashStyle.Dash;
       this.CreateGraphics().DrawLine( coloredPen, 320, 30, 395, 150 );
           //-------------------------------------------------------------------
           // pen and location for red outline rectangle
            Pen thickRedPen = new Pen( Color.Red, 10 );
            this.CreateGraphics().DrawRectangle(thickRedPen, 80, 30, 65, 100 );

           //-------------------------------------------------------------------
           // ellipse rectangle and gradient brush
          Rectangle drawArea1 = new Rectangle( 5, 35, 20, 20 );
         LinearGradientBrush linearBrush1 = new LinearGradientBrush(drawArea1,
                  Color.Red,Color.Yellow, LinearGradientMode.ForwardDiagonal);
```

```
        Rectangle drawArea2 = new Rectangle(5, 35, 10, 10);
        LinearGradientBrush linearBrush2 = new LinearGradientBrush(drawArea2,
                  Color.Blue, Color.Yellow, LinearGradientMode.ForwardDiagonal);
          //---------------------------------------------------------------------

          // draw ellipse filled with a blue-yellow gradient
        this.CreateGraphics().FillEllipse(linearBrush2, 5, 30, 65, 100);
        this.CreateGraphics().FillRectangle(linearBrush1, 155, 30, 75, 100);
        this.CreateGraphics().FillRectangle(linearBrush2, 450, 30, 75, 100);


      } // end btn
    } // end class
} // end namespace
```




the end

The End

The End

The End

The End

The End

The End

The End

